

INTERSIL

\$5.00

INTERCEPT® JR. MICROCOMPUTER TUTORIAL SYSTEM USER'S MANUAL



Rev. B-August, 1979

TABLE OF CONTENTS

CHAPTER		PAGE
1	INTRODUCTION	1-1
2	WORKING WITH THE INTERCEPT JR. MODULE	2-1
	INTERCEPT JR. START-UP	2-1
	RESET SWITCH	2-2
	ENTERING THE CONTROL MODE	2-2
	SELECTING A FUNCTION	2-2
	INSPECT ACCUMULATOR	2-3
	SETPC	2-3
	DECREMENT PC, DECPC	2-3
	DEPOSIT DATA INTO MEMORY, MEM	2-3
	RUN	2-4
	HALT	2-4
	RESET	2-4
	SIN	2-5
	DIS	2-5
	BIN LOADER	2-5
	MICRO	2-5
	MICROINTERPRETER FUNCTIONS	2-5
	MEMORY REFERENCE INSTRUCTIONS	2-6
	INPUT/OUTPUT TRANSFER (IOT) INSTRUCTIONS	2-6
	OPERATE INSTRUCTIONS	2-6
	LEAVING MICRO MODE	2-7
	PROGRAM EDITING AND CORRECTION	2-7
	KEYPAD OPERATION	2-8
	TABLE OF INSTRUCTION CODES	2-10
3	INTERCEPT JR. PROGRAMMING EXAMPLES	3-1
	EXAMPLE 1 - INCREMENTING MEMORY DATA	3-1
	EXAMPLE 2 - DECREMENTING MEMORY DATA	3-1
	EXAMPLE 3 - PROGRAMMING TIME DELAYS	3-2
	EXAMPLE 4 - ADDRESSING MODES	3-2
	EXAMPLE 5 - INDIRECT ADDRESSING USED IN TABLE MANIPULATION	3-3
	EXAMPLE 6 - THE JMS INSTRUCTION AND INDIRECT ADDRESSING	3-5
	EXAMPLE 7 - AUTOINDEXING	3-6
	EXAMPLE 8 - ADDRESS FIELD MODIFICATION	3-7
	EXAMPLE 9 - USING CONDITIONAL SKIPS	3-8
	EXAMPLE 10 - FLOWCHARTING A PROGRAM	3-10
	EXAMPLE 11 - BIT MANIPULATION	3-14
	EXAMPLE 12 - LOGICAL OPERATIONS	3-15
	EXAMPLE 13 - I/O PROGRAMMING	3-15
	EXAMPLE 14 - TELETYPE I/O USING MONITOR CALLS	3-17
	EXAMPLE 15 - PRINTING UNDER KEYPAD CONTROL	3-18
	EXAMPLE 16 - PROGRAM TO DEMONSTRATE I/O TO THE 6957 AUDVIS MODULE	3-19
	EXAMPLE 17 - REAL-TIME PROGRAMMING	3-20

CHAPTER		PAGE
4	INTERCEPT JR. MODULE	4-1
	INTRODUCTION	4-1
	TYING ON TO THE DX BUS	4-1
	ADDRESS DEMULTIPLEXING	4-2
	DATA DEMULTIPLEXING	4-2
	KEYBOARD INPUT	4-3
	DIGITAL DISPLAY OUT	4-3
	IOT PROCESSING	4-4
	OPTIONS	4-7
	SCHEMATIC	4-8
5	JR. RAM MODULE	5-1
	INTRODUCTION	5-1
	DISCUSSION	5-1
	SCHEMATIC	5-4
6	JR. P/ROM MODULE	6-1
	INTRODUCTION	6-1
	DISCUSSION	6-2
	SCHEMATIC	6-4
7	JR. PIEART SERIAL I/O MODULE	7-1
	INTRODUCTION	7-1
	DISCUSSION	7-1
	SCHEMATIC	7-9
8	INTERCEPT JR. TUTORIAL SYSTEM MONITOR PROGRAM	8-1
	INTERCEPT JR. MAIN FLOWCHART	8-2
	MONITOR STACK	8-15
	REFSH - DISPLAY REFRESH	8-18
	SEDB - SWITCH DEBOUNCE	8-20
	CLKPD - CLEAR KEYPAD	8-21
	HEX	8-21
	EXIT	8-21
	CONTROL STATE SERVICE ROUTINES	8-21
	INCAC - INCREMENT ACCUMULATOR	8-21
	DEPCPC - DECREMENT PROGRAM COUNTER	8-21
	HALT	8-21
	RUN	8-21
	RESET	8-22
	DEPOSIT INTO MEMORY	8-22
	BLANK FLAG TOGGLE	8-23
	SETPC - SET PROGRAM COUNTER	8-23
	MICRO - MICROINTERPRETER	8-24
	SIN - SINGLE INSTRUCTION EXECUTE	8-27
	INPIE - INITIALIZE PIE	8-29
	BIN - BINARY LOADER	8-29
	DUMP - MEMORY DUMP	8-29
	MONITOR PROGRAM LISTING	8-31
9	INTERCEPT JR. AUDIO CARD	9-1
	INTRODUCTION	9-1
	DISCUSSION	9-2
	SCHEMATIC	9-3

CHAPTER		PAGE
10	INTERCEPT JR. CASSETTE INTERFACE CARD	10-1
	INTRODUCTION	10-1
	THE RECORDER	10-4
	SOURCES OF NOISE INTERFERENCE	10-5
	6954 ACI BOARD	10-6
	WRITING PROGRAMS FOR DATA TRANSFERS - USING TTY ADDRESSES	10-7
	USER ASSIGNED ADDRESS	10-8
	WRITING A MEMORY WORD TO THE CASSETTE	10-8
	READING A MEMORY WORD FROM THE CASSETTE	10-9
	INITIALIZING THE PIE	10-9
	THEORY OF RECEIVER SECTION OPERATION	10-10
	THEORY OF TRANSMITTER SECTION OPERATION	10-11
	D.C.-D.C. CONVERTER	10-12
	CIRCUIT OPTIONS	10-12
	SCHEMATIC	10-13

APPENDICES

APPENDIX A	INTERCEPT JR. PROGRAMMING FUNDAMENTALS	A-1
	NUMBERING SYSTEMS	A-1
	ARITHMETIC PROGRAMMING EXAMPLE #1	A-6
	ARITHMETIC PROGRAMMING EXAMPLE #2	A-9
	ARITHMETIC PROGRAMMING EXAMPLE #3	A-10
	BINARY & OCTAL ADDITION & MULTIPLICATION TABLES	A-12
APPENDIX B	INTRODUCTION TO LOGIC	B-1
APPENDIX C	OCTAL-DECIMAL INTEGER CONVERSION TABLE	C-1
APPENDIX D	INSTRUCTION SUMMARY AND BIT ASSIGNMENTS	D-1
APPENDIX E	GLOSSARY	E-1
APPENDIX F	ASCII CHARACTER CODES	F-1
APPENDIX G	LOADING CONSTANTS INTO THE ACCUMULATOR	G-1
APPENDIX H	OPERATION OF THE PHASELOCK LOOP	H-1
APPENDIX I	IM6100 CMOS MICROPROCESSOR REMOTE DATA STATION	I-1
APPENDIX J	KEYBOARD TENNIS PROGRAM WITH INTERCEPT JR.	J-1
APPENDIX K	OCTAL DEBUGGING TECHNIQUE ROM	K-1
APPENDIX L	INTERSIL ODT LISTING	L-1

FIGURES

<u>FIGURE</u>		<u>PAGE</u>
1-1	INTERCEPT JR. TUTORIAL SYSTEM	1-2
1-2	INTERCEPT JR. SYSTEM BLOCK DIAGRAM	1-4
2-1	INTERCEPT JR. MODULE	2-1
2-2	MEMORY REFERENCE INSTRUCTION FORMAT	2-11
2-3	IOT INSTRUCTION FORMAT	2-12
2-4	GROUP 1 MICROINSTRUCTION FORMAT	2-15
2-5	GROUP 2 MICROINSTRUCTION FORMAT	2-17
2-6	GROUP 3 MICROINSTRUCTION FORMAT	2-18
3-1	PROGRAM FLOWCHART	3-11
5-1	JR. RAM MODULE	5-1
6-1	JR. P/ROM MODULE	6-1
7-1	JR. SERIAL I/O MODULE	7-1
8-1	MEMORY ALLOCATION MAP	8-1
8-2	INTERCEPT JR. MAIN FLOWCHART	8-2
8-3A	STATUS WORD LOCATION 0143	8-18
8-3B	SWITCH WORD LOCATION 0133	8-19
8-3C	ACTIVE DISPLAY OPTIONS	8-19
8-3D	DISPLAY FORMATTING	8-20
8-4	CONTROL STATE KEY SELECTION/CONNECTIONS KEY - DX BUS	8-17
8-5	MONITOR PROGRAM LISTING	8-31
9-1	INTERCEPT JR. AUDIO CARD	9-1
10-1	INTERCEPT JR. CASSETTE INTERFACE CARD	10-1

TABLES

2-1	TABLE OF INSTRUCTION CODES	2-10
3-1	PAGE VS. MEMORY LOCATIONS	3-5
5-1	JUMPER CONNECTIONS FOR MAPPING	5-2
6-1	ADDRESS RANGE IN OCTAL IM5623/IM5624	6-2
7-1	CONTROL REGISTER A CONSTANTS	7-2
7-2	CONTROL REGISTER B CONSTANTS	7-3
7-3	VECTOR REGISTER	7-3
7-4	UART CONTROL REGISTER BIT	7-4
7-5	20 mA LOOP/EIA RS232-C CONNECTOR PINOUTS	7-6
7-6	PIE-UART INSTRUCTIONS	7-7

CHAPTER 1

INTRODUCTION

The theoretical principles underlying digital computers were first enunciated by Charles Babbage in 1833, but the technology available at the time was not equal to the task of actually building a working machine. John Von Neumann developed the stored program concept at the Institute for Advanced Studies at Princeton University and, since then, electronic computers have undergone several iterations from the early vacuum tube machines to transistorization to integrated circuit systems, and now the age of LSI is evident. Architectural advances from the first use of hardware index registers, micro-programmed control, interrupt processing, direct memory access channels, and distributed processing have been numerous, but the history of digital computers has yet to be fully written.

In the last 1930's and early 1940's, wartime requirements and the development of vacuum tubes led to the construction of extremely expensive and complex digital computers used mainly to speed up numerical calculations. As the technology progressed, computers became faster, smaller and less expensive. Advances in hardware architecture and programming languages evolved rapidly. As a result, the 1960's saw significant increases in the application of business and data processing computers.

The first minicomputer, the PDP-8*, was introduced by Digital Equipment Corporation in 1965 and made dedicated applications for digital computers possible. This first minicomputer, costing approximately \$50,000, was considered so inexpensive that it found itself being used in universities, laboratories, and in numerous process control applications. Many versions of this machine were brought out in succeeding years.

Computers, big and small, must all have a processor, main memory and input/output. Decreasing hardware costs and increasing sophistication of processing technology led to multiplicity of computer architecture. The early 1970's saw the microprocessor, the heart of a computer, enter the scene. Its function is to accept data from the user, process it according to instructions provided by the user, and store in memory, and return usable results to the user in some convenient fashion.

LSI techniques, with their high density capability, have enabled semiconductor manufacturers to produce processing units and memory devices on single monolithic silicon chips. Input and output devices which constitute the man/machine interface, have remained relatively bulky.

* Trademark of Digital Equipment Corporation, Maynard, Mass.

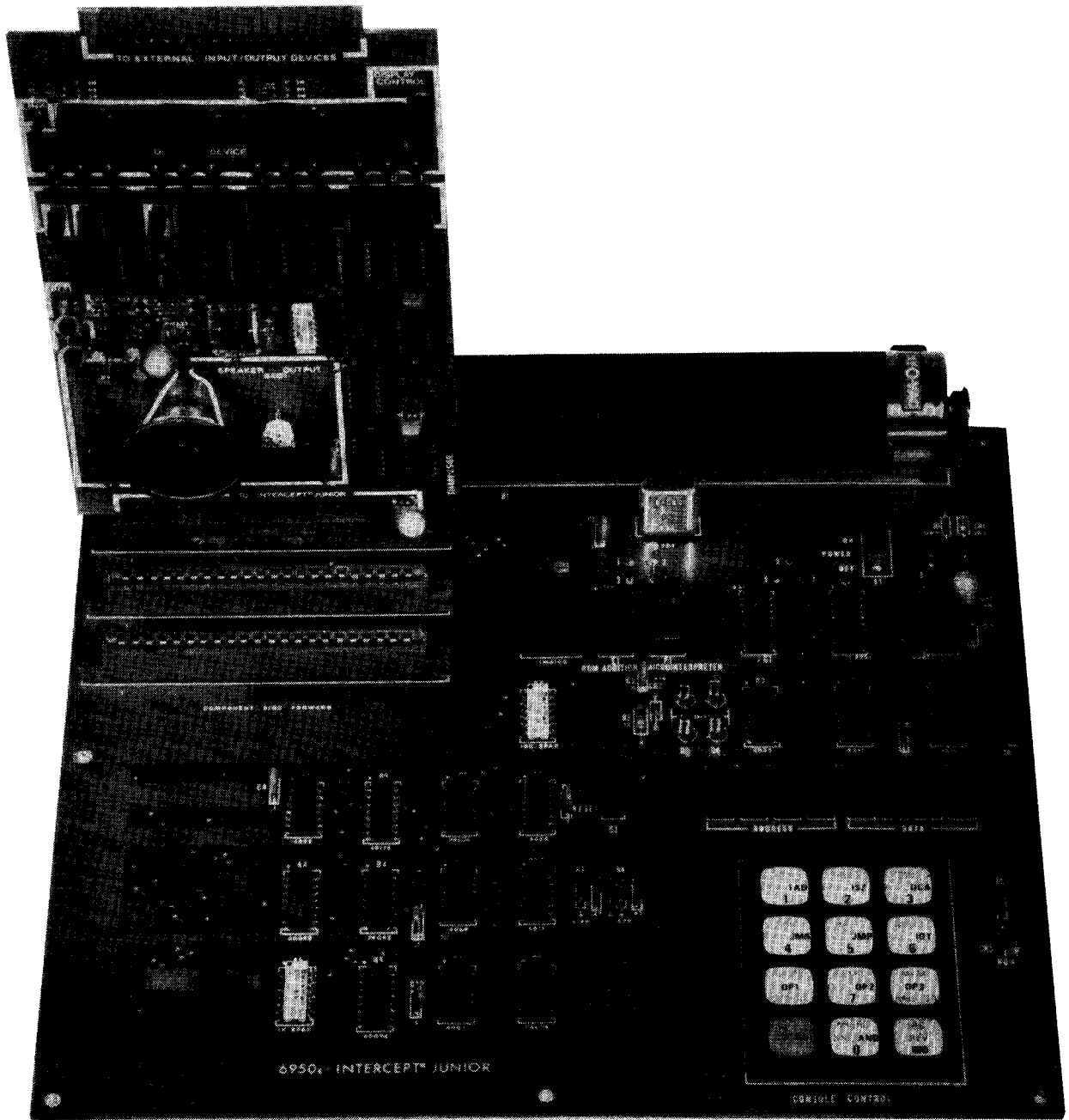


FIGURE 1-1

The INTERCEPT JR. TUTORIAL SYSTEM, pictured in Figure 1-1, recognizes the instruction set of Digital Equipment Corporation's PDP-8/E and is designed with a modular concept to enable the user to purchase only those modules which meet his requirements. The design permits the user to participate in the future of digital computers by yielding an understanding of the microprocessor and related component functions as well as programming fundamentals.

Large Scale Integration (LSI) of Intersil's digital CMOS components results in the system being battery operable and, thereby, yields the flexibility of a portable system. Experience can be gained with the components required for a classical computer architecture--a processor, or central processing unit (CPU), memory and input/output. The IM6100 microprocessor serves as the CPU and memory is available in the form of CMOS RAM, ROM and bipolar P/ROM. Input/output can be experienced in its simplest form via the keyboard and LED displays or can be studied in greater detail by utilizing the JR. SERIAL I/O MODULE.

This Owner's Handbook presents a step-by-step learning experience for the INTERCEPT JR. TUTORIAL SYSTEM. Chapter 2 entitled "Working With the Intercept Jr. Module" instructs the user in the fundamentals of the basic module--the start-up and the selection of a function. The console control, or keyboard, is discussed in detail. Chapter 3, "Programming Fundamentals", presents the user with simple programming examples and the ability to progress to more complex problems. Chapter 4, 5, 6, 7, 9 and 10 explain the hardware aspects of the six modules via pictorial representation, text and the corresponding schematics. Chapter 8 discusses the monitor ROM program, presents the flow chart and listing, and, thereby, gives the user a greater degree of programming insight. The Appendices contain fundamental information on number systems, two's complement arithmetic, an introduction to logic, and other miscellaneous information that will be of interest to the user. Figure 1-2 presents a block diagram of the total system configuration.

It is Intersil Incorporated's opinion that the INTERCEPT JR. TUTORIAL SYSTEM will enable you to embark upon a truly rewarding educational experience. The microprocessor has resulted in a natural evolutionary step in electronic circuitry design. This is only the beginning. We sincerely wish that your participation in this evolution will be rewarding to you.

SYSTEM BLOCK DIAGRAM

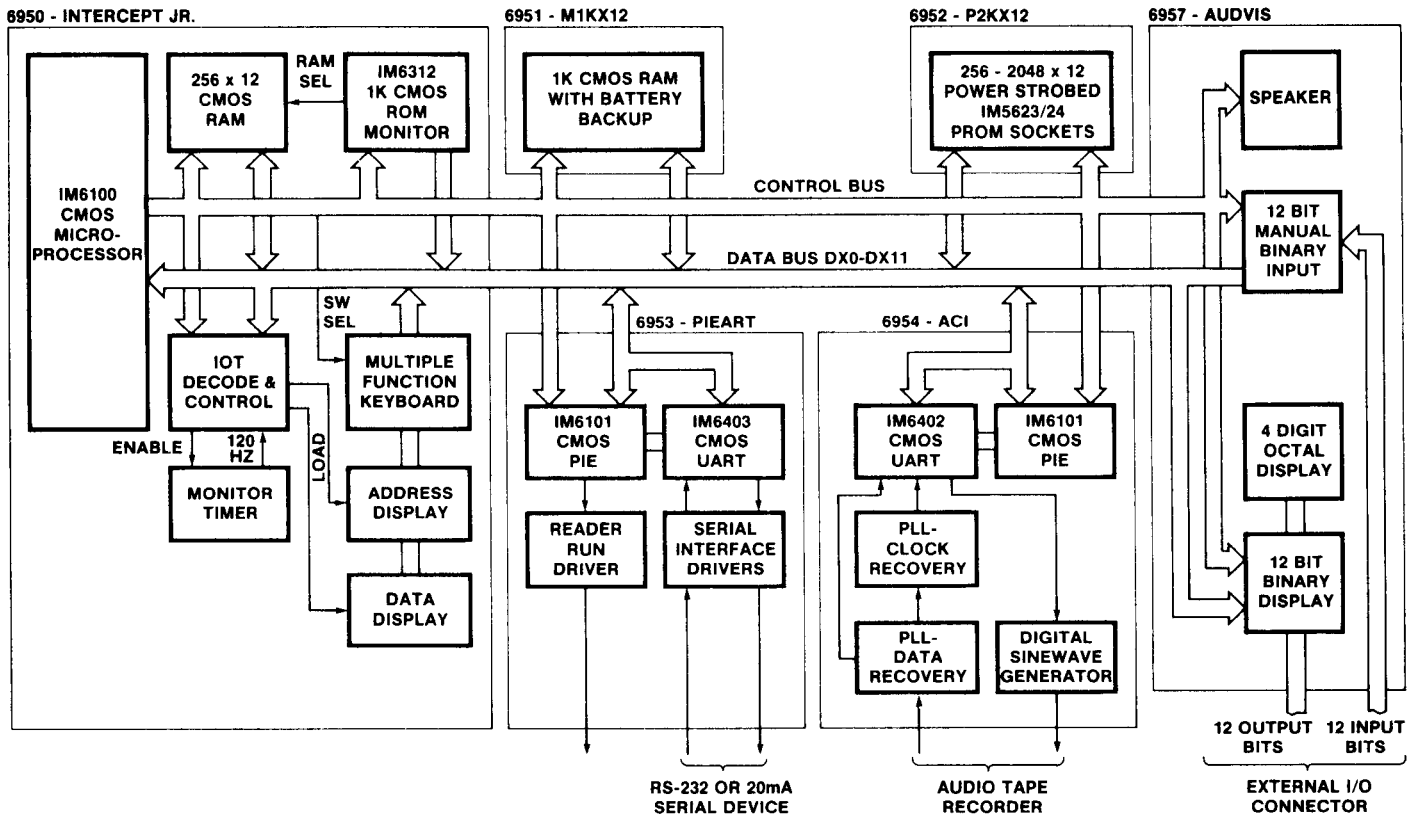


Figure 1-2

CHAPTER 2
WORKING WITH THE INTERCEPT JR. MODULE

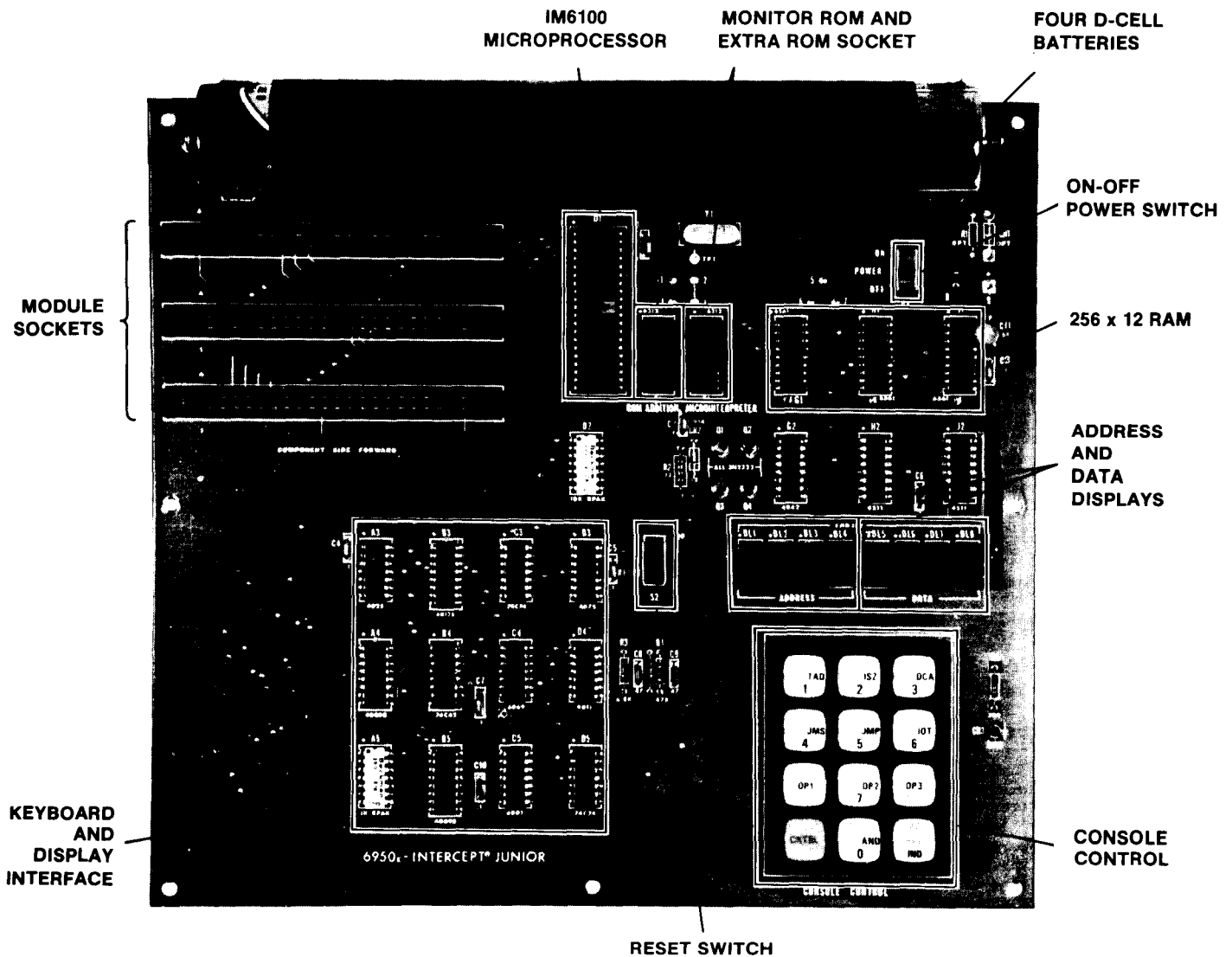



Figure 2-1 provides a pictorial representation of the INTERCEPT JR. MODULE with the pertinent components discussed in this chapter highlighted.

INTERCEPT JR. START-UP

Turn the module "ON" with the "ON-OFF" power switch. Power is provided by the four (4) D-Cell batteries which must be inserted, with the sleeve, in the module. When facing the module, with the keyboard in front and connectors on the left side, the left hand

battery clip is negative and the right hand battery clip is positive. BATTERY REVERSAL WILL DAMAGE THE SYSTEM. The module does a power-up RESET so that it will always come up halted with the Program Counter, PC (ADDRESS) equal to 7777. the CONSOLE CONTROL timer will be active so that the ADDRESS and MEMORY displays will be valid provided a "BLANK DISPLAY" is not in effect. The information displayed will be PC = 7777 in the ADDRESS and the MEMORY data in that location will be 5776. This instruction branches the microprocessor to routines which save registers, initialize the RAM stack and search for keyboard depressions. If the display does not illuminate, press  to turn it on. The 256 words of RAM are always provided power, as long as the batteries are installed, regardless of the position of the "ON-OFF" power switch.

RESET SWITCH

The RESET SWITCH does a complete hardware reset of the microprocessor and can be used at any time for this purpose. Therefore, it is not necessary to turn power off to reset the microprocessor. When switching the module OFF it is recommended that the RESET SWITCH be slid down while the power is turned off. This keeps the microprocessor from running during the power down process thereby eliminating the possibility of writing bad data into the RAM as the voltage level goes lower than the minimum specified. If the RAM data is not required to be preserved, use of the RESET SWITCH is not required during power-off.

ENTERING THE CONTROL MODE



The operator will now enter the control mode by pressing the red control key, CNTRL, on the KEYBOARD. This key will cause the module to enter what is referred to as an undefined control mode or SHELL mode when the CONSOLE CONTROL timer is enabled, or at any point during the execution of a control function. This state is referred to as undefined as we have not yet chosen a CONSOLE CONTROL function to be performed. The other mode is the user mode in which the module is either waiting for or executing user programs.

SELECTING A FUNCTION

After pressing the CNTRL key, we are now ready to choose a function to be performed. This is accomplished by pressing any of the function keys which are described next.

INSPECT ACCUMULATOR



Pressing CNTRL IAC will change the mode of the right hand display from memory data to accumulator contents or vice versa. If bit 7 of the SWITCH word has been zeroed (see chapter 8), instead of the AC, the contents of any location may be continuously displayed during program execution. This key also has special meaning for certain functions and its use is described with each of those functions. This key is color-coded yellow.

SETPC



This function allows the user to control the Program Counter, PC, in the module for purposes of depositing words, or examining words or conditions. Following the activation of this key, the user will load an octal number into the PC by entering the digits on keys 0-7. The digits will be displayed in ADDRESS and will be entered from the right, shifting the previously entered digits to the left. Any number of digits may be entered until the display contains the value desired. A CNTRL key depression will enter the value displayed into the PC and will return the state to SHELL mode. Note that leading zeros may be needed to clear the display before entering the desired octal numbers.

DECREMENT PC, DECPC



This function will decrement the value of the PC by one and return the module to SHELL mode. This function is useful when examining sequences of memory locations.

DEPOSIT DATA INTO MEMORY, MEM



This function allows the user to enter instructions and data in the RAM (see Figure 8-1) as well as set the values of the internal registers of the module by depositing the data into memory locations used by the monitor program to save and update the data in these registers. After a closure of the MEM key, the user will proceed to enter digits into MEMORY with keys 0-7 as he did for SETPC. The new digits will be displayed on MEMORY, entering from the right and shifting to the left. When the MEMORY display contains the desired value, the user will deposit it in the RAM by pressing either DECPC, or MEM. If

DECPC is pressed, the MEMORY display will be deposited into RAM in the memory location addressed by the ADDRESS display. The ADDRESS display will be decremented and the RAM information in the decremented address will be displayed in MEMORY. If MEM is pressed, the value shown in the MEMORY display will be deposited into the RAM in the memory location addressed by the ADDRESS display, the ADDRESS display will be incremented and the next word in RAM will be displayed in MEMORY. Successive depressions of MEM will increment the memory ADDRESS. Digits can now be entered from the right, as before. If the user wishes to skip a location, he presses MEM again. This will retain the value of that location in RAM and the ADDRESS will move to the next location. By pressing the yellow key, the user will deposit the value of MEMORY into the location specified by ADDRESS, the module will exit the control mode and enter the user mode. If the user presses CNTRL, the value shown in MEMORY will be deposited and the module will enter the SHELL mode. RAM locations 0000 and 0143-0177, reserved for the MONITOR cannot be modified. Locations 0140-0142, also used by the MONITOR, should not be modified ordinarily.

RUN



This function will set the microprocessor Run flip flop to RUN and will exit the control mode. The module will come out in the user mode at the PC point specified during control mode, running.

HALT



This function will clear the RUN flip flop in the microprocessor so that the module will come out of the control mode halted.

RESET



This function will be a complete software RESET of the module. All internal microprocessor flags are initialized, the accumulator and link are cleared, the PC is set to 0200 and its contents are displayed. It will also remove a BLANK DISPLAY status.

SIN



This function, referred to as Single Instruction, will cause the module to perform, in the user mode, a single instruction. Following this, the possible changes of state can be observed by inspecting the contents of the appropriate memory locations. Due to the MONITOR program structure, the user cannot single step through ROM-P/ROM locations. JMP*-1, JMS*-1 and JMS*-2 instructions can be single stepped properly; but TAD, ISZ and DCA instructions which refer to a *+1 or *+2 location cannot be single stepped properly (see Chapter 8). SIN may be successively depressed to single step through a program.

DIS



This function will BLANK and RESTORE the ADDRESS and MEMORY display thereby conserving power. The BLANK/RESTORE function is achieved by depressing CNTRL followed by DIS to BLANK the display and then CNTRL followed by DIS to RESTORE the display. A blanked display will carry over from a power-down but will be cleared by a software RESET (depression of CNTRL and RESET). The RESET switch does not affect display status.

BIN LOADER



This function will activate the firmware loader which will load BINary tapes using the 6953-PIEART, JR. SERIAL I/O MODULE. This loader will return to the halted user mode when data has finished loading. The BIN loader will ignore data for locations 0000g and 0143g-0177g and will load all BIN formatted tapes generated by the MONITOR or by PDP-8/E or IM6100 assemblers. The loader will ignore all change field instructions on those tapes. It will also echo all characters enclosed by rubouts on those tapes.

MICRO



This function will place the CONSOLE CONTROL at the control of the MICROINTERPRETER in the MONITOR ROM. The MICROINTERPRETER functions are elaborated on in the next section.

MICROINTERPRETER FUNCTIONS

Pressing CNTRL followed by MICRO causes INTERCEPT JR. to execute the microinterpreter routines which are resident in the MONITOR

ROM. These routines will interpret key closures as opcode bits, relative address bits, page bits, address mode bits, and micro-instruction bits according to the specific sequence in which the keys are depressed. This enables the user to rapidly enter programs via the keyboard without constantly referring to the instruction format listings. The user should be familiar with the use of the instructions in order to make the most efficient use of the microinterpreter.

MEMORY REFERENCE INSTRUCTIONS

In the MICRO mode, if any of the keys marked AND, TAD, ISZ, DCA, JMS, or JMP are pressed, the MEMORY display at the current memory ADDRESS will show 0000, 1000, 2000, 3000, 4000, or 5000, respectively.

All the following key closures are interpreted as address bits. The numerical keys may be depressed as many times as desired, entering octal address digits from right to left. While address digits are being entered, the opcode will be displayed on the left hand display.



At any time after the opcode is entered, depression of the yellow IAC key will set the indirect bit (note IND legend on this key) of the instruction (add 4 to the next-to-most significant octal digit). After entering the address, depressing CNTRL will display the ADDRESS counter and the fully assembled instruction. Releasing CNTRL will advance the ADDRESS counter. The yellow IAC key may be pressed repeatedly to advance the ADDRESS counter.

INPUT/OUTPUT TRANSFER (IOT) INSTRUCTIONS



In the MICRO mode, depression of the IOT key will cause 6000 to be entered into the currently addressed memory location. Subsequent numeric key depressions are performed to enter the required device address and control bits into the IOT instruction.

Depressing CNTRL will advance the ADDRESS counter to the next location. Depressing SHIFT will cause the ADDRESS counter to step.

OPERATE INSTRUCTIONS

Operator instructions are divided into three groups of operate microinstructions. Thus, in the MICRO mode, the desired microinstruction group is selected by depressing the keys marked OPR1, OPR2 or OPR3. This will enter 7000, 7400 or 7401, respectively, into the MEMORY display.

If no additional keys are depressed and the address counter is advanced, these instructions, which are all NO OPERATION, NOP, will be entered. Further key depressions will set various bits in the instruction enabling the user to select valid microinstruction combinations. The microinterpreter does not check for illegal microinstruction combinations so the user must be careful about the combinations being selected. Table 2-1 shows the more useful combinations. The user should become familiar with the rules of combinations and logical execution sequence in order to create microinstructions not shown in the tables.



On the CONSOLE CONTROL, in general, the designations in red are associated with OPR1 microinstructions, and the designations in green, except for -QA and -QL, are associated with OPR2 microinstructions. The -QA and -QL designations stand for MQA and MQL which are OPR3 microinstructions.



Conditional skip microinstructions in the OPR2 group may have their skip condition inverted by pressing the REV key while setting the microinstruction bits.



Rotate instructions in the OPR1 group may be changed from a single bit rotate to a two bit rotate by pressing the key with T/BSW designation on it. (This key is used for both two bit rotates as well as Byte SWap.)

LEAVING MICRO MODE

Depressing the CNTRL key twice puts the user back into SHELL, the undefined control state, and free to choose the next function.

PROGRAM EDITING AND CORRECTION

If an instruction is entered incorrectly, the user must exit MICRO by depressing CNTRL twice. This will result in advancing the ADDRESS counter by one. Decrementing the ADDRESS counter by one is achieved by pressing DECPC. The user must then reenter the MICRO mode by pressing CNTRL and MICRO. Now the correct instruction may be reentered in full.

The program may be examined location by location by successively pressing DECPC or MEM from the undefined control state. DECPC results in stepping backward through memory, and MEM results in stepping forward. These two keys may be pressed without going through the undefined control state in order to go backwards and forwards through the program in any sequence.

Memory data may be changed at will while stepping back and forth through the program simply by depressing the numeric keys in any desired fashion.

When editing in the MICRO mode, an instruction may be changed by entering a new sequence of keys. If an instruction is correct, the address counter may be stepped simply by pressing the yellow IAC key (immediately after CNTRL has been pressed to step the address) as many times as desired.

KEYPAD OPERATION

We shall illustrate keypad operations in MICRO mode with this example:

Enter instruction JMP START in location 0357₈.
Label START represents location 0200₈. Enter instruction SZL in location 0362₈.

- 1) To set program counter to 0357₈, press CNTRL SETPC
0 3 5 7

Comments: Note that the same key has the digit 7 and the legend SETPC on it. Thus, the MONITOR routines assigned different meanings to the keys at different times. The address is shifted from right to left into the left hand display.

- 2) MICROINTERPRETER mode, press CNTRL MICRO

Comments: When CNTRL is pressed, the SETPC mode is terminated and when MICRO is pressed, the MICRO mode is entered.

- 3) OPCODE entry, press JMP

Comments: The digit 5 appears in the most significant position of the right display and the other positions are clear. 0357 5000

- 4) ADDRESS entry, press 0 2 0 0

Comments: As soon as key 0 is pressed, the display switches to 5000 0000 and any string of octal digits may be entered into the right display from right to left.

- 5) ASSEMBLE complete instruction, place in memory and increment program counter, press CNTRL.

Comments: Note that if the yellow key is pressed at this point, the MICROINTERPRETER will set the indirect bit, that is, add 0400 to the opcode and the location referenced by the instruction will be used as a pointer to the effective address.

As CNTRL is pressed and held down, the displays will show the PC and assembled instruction 0357 5200, and when the key is released, the left display will increment to 0360g and the MICRO mode is again in effect waiting for another opcode entry. At this point, if the key marked IAC REV IND is pressed, the address will increment again with MICRO still in control. If the address that was entered was not in page 0 (0000g to 0177g) or in the current page (0200g to 0377g), a simple diagnostic message consisting of a flashing display is received by the user and another attempt to enter a valid address may be made.

- 6) ADVANCE address to 0362g, press IAC IAC IAC

Comments: This is quicker than pressing CNTRL SETPC 0 3 6 2 CNTRL MICRO.

- 7) Enter SZL instruction (skip on zero link), press OPR2 SNL REV

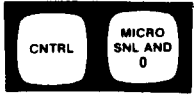
Comments: By pressing OPR2, bits 0, 1, 2, 3 are set showing 7400g in the right display. By pressing SNL (the same key that previously was used to enter the MICRO mode), bit 7 is set showing 7420g in the right display, the code for an SNL. Pressing REV (the same key with IAC on it) reverses the skip condition from non-zero to zero in this case by setting bit 8, and 7430g is seen in the right display.

This example shows how the MICROINTERPRETER assigns multiple meanings to the keys. The twelve keys of the keypad are read onto the 12-bit micro-processor data bus under program control.

TABLE 2-1

TABLE OF INSTRUCTION CODES

KEYS DEPRESSED
LEFT TO RIGHT



MEMORY
OCTAL CODE

-

OPERATION

Enter MICROINSTRUCTION Mode

MEMORY REFERENCE INSTRUCTIONS

KEYS DEPRESSED
LEFT TO RIGHT



MNEMONIC

MEMORY
OCTAL CODE

OPERATION

AND*

0000

Logical AND

nn....n

00nn or
01nn

Depress numeric keys as
required for valid address



04nn or
05nn

Depress IND key if INDirect
MRI is required



Advances ADDRESS counter to
next location



TAD

1000

Binary ADD



ISZ

2000

Increment and Skip if Zero

* The sequence of key depressions required to enter the opcode, address field, indirect bit (if necessary) and advance the address counter is shown in full for this case. The same sequence is true for the other memory reference instructions, but only the initial operation of entering the opcode is shown for the remainder to avoid duplication.



DCA 3000 Deposit and Clear Accumulator



JMS 4000 JuMp to Subroutine



JMP 5000 JuMP

MEMORY REFERENCE INSTRUCTION FORMAT

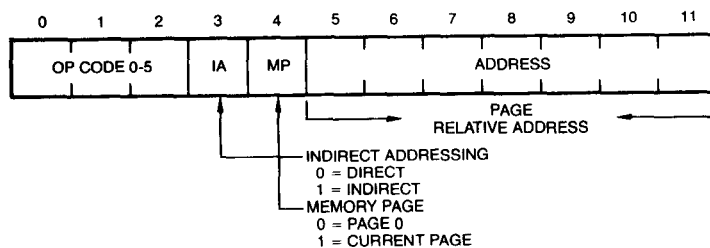


FIGURE 2-2

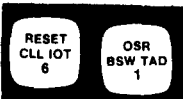
MICROPROCESSOR INPUT/OUTPUT TRANSFER (IOT) INSTRUCTIONS

KEYS DEPRESSED
LEFT TO RIGHT

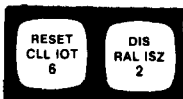


MNEMONIC MEMORY OCTAL CODE OPERATION

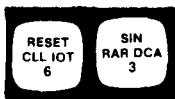
SKON 6000 Skip if Interrupt on



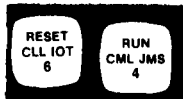
ION 6001 Interrupt Turn on



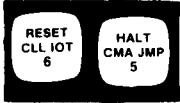
IOT 6002 Interrupt Turn off



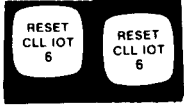
SRQ 6003 Skip if INT Request



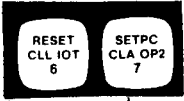
GTF 6004 Get Flags



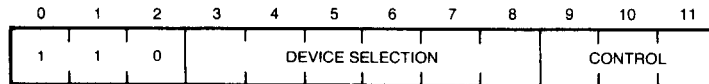
RTF 6005 Return Flags



SGT 6006 Operation is Determined by External Device, if Any



CAF 6007 Clear All Flags



IOT INSTRUCTION FORMAT

FIGURE 2-3

DEVICE INPUT/OUTPUT TRANSFER (IOT) INSTRUCTION

KEYS DEPRESSED LEFT TO RIGHT

MNEMONIC

MEMORY OCTAL CODE

OPERATION



As applicable

6000

n, n, n, n...n

6nnn

Depress numeric keys as required to enter specific address and control bits

GROUP 1 OPERATE MICROINSTRUCTIONS

KEYS DEPRESSED LEFT TO RIGHT

MNEMONIC

MEMORY OCTAL CODE

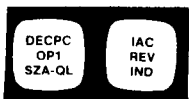
OPERATION



NOP

7000



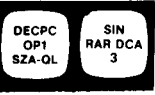

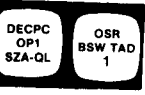
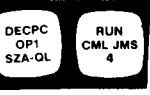
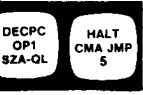

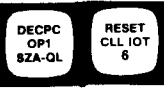

No operation

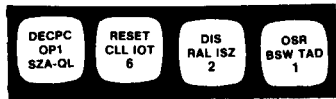


IAC

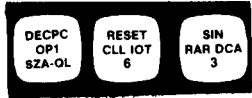
7001

Increment Accumulator

	RAL	7004	Rotate Accumulator Left
	RTL	7006	Rotate Two Left The T in T/BSW indicates bit 10 is set to give two shifts. Key may be pressed before RAL if desired.
	RAR	7010	Rotate Accumulator Right
	RTR	7012	Rotate Two Right Except for OPR1 key, order of depression is irrelevant.
	BSW	7002	Byte Swap Only bit 10 set giving byte swap function.
	CML	7020	Complement Link
	CMA	7040	Complement Accumulator
	CIA	7041	Complement and Increment Accumulator Logical execution sequence is CMA, IAC, but keys may be pressed in IAC, CMA order.
	CLL	7100	Clear Link
	CLL RAL	7104	Clear Link-Rotate Accumulator Left Logical sequence first clears link, then rotates.



CLL RTL 7106 Clear Link-Rotate Two Left



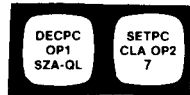
CLL RAR 7110 Clear Link-Rotate Accumulator Right



CLL RTR 7112 Clear Link-Rotate Two Right



STL 7120 Set the Link Logical sequence first clears, then complements link.



CLA 7200 Clear Accumulator Common to all groups, so not colored.



CLA IAC 7201 Clear Accumulator-Increment Accumulator Loads accumulator with 1.



GTL 7204 Get the Link Accomplished by rotating it into cleared accumulator.



CLA CLL 7300 Clear Accumulator-Clear Link

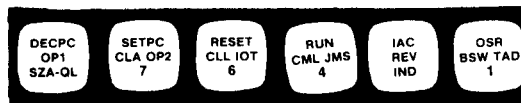


STA 7240 Set the Accumulator Sets accumulator to all ones.

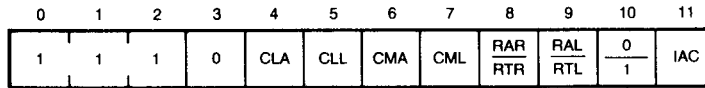
Example of microprogrammed instruction to set accumulator to octal six.

Logical sequence: CLA CLL CML IAC RTL

Key sequence:



Octal instruction: 7327



BSW IF BITS
8 & 9 ARE 0
AND BIT 10 IS 1.

LOGICAL SEQUENCES:
1—CLA, CLL
2—CMA, CML
3—IAC
4—RAR, RAL, RTR, RTL, BSW


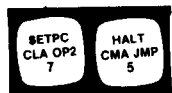
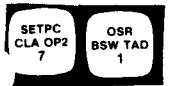
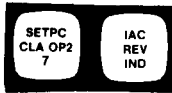
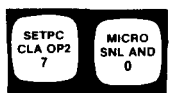

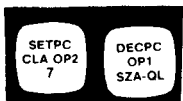
GROUP 1 MICROINSTRUCTION FORMAT



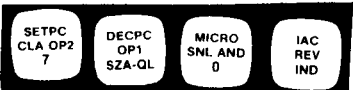
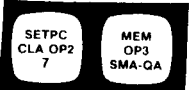


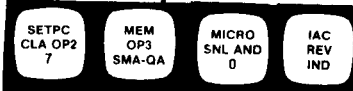

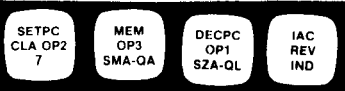
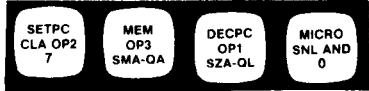
FIGURE 2-4

GROUP 2 OPERATE MICROINSTRUCTIONS

KEYS DEPRESSED
LEFT TO RIGHT

MNEMONIC MEMORY
 OCTAL CODE OPERATION

	NOP	7400	No operation
	HLT	7402	Halt
	OSR	7404	Or with Switch Register
	SKP	7410	Skip REV key sets bit 8 giving the AND condition of skips specified in bits 5, 6, 7. This results in unconditional skip.
	SNL	7420	Skip on Non-Zero Link
	SZL	7430	Skip on Zero Link REV reverses selected skip condition by setting bit 8.
	SZA	7440	Skip On Zero Accumulator

	SNA	7450	Skip on Non-Zero Accumulator
	SZA SNL	7460	Skip on Zero Accumulator, or Skip on Non-Zero Link, or both OR'ed skip conditions.
	SNA SZL	7470	Skip on Non-Zero Accumulator, and Skip on Zero Link AND'ed skip conditions.
	SMA	7500	Skip on Minus Accumuator
	SPA	7510	Skip on Positive Accumulator
	SMA SNL	7520	Skip on Minus Accumulator, or Skip on Non-Zero Link, or both OR'ed skip conditions.
	SPA SZL	7530	Skip on Positive Accumulator and Skip on Zero Link AND'ed skip conditions.
	SMA SZA	7540	Skip on Minus Accumulator or Skip on Zero Accumulator or both. OR'ed skip conditions.
	SPA SNA	7550	Skip on Positive Accumulator and Skip on Non-Zero Accumulator AND'ed skip conditions.
	SMA SZA SNL	7560	Skip on Minus Accumulator or Skip on Zero Accumulator or Skip on Non-Zero Link or all OR'ed skip conditions.

	SPA SNA SZL	7570	Skip on Positive Accumulator and Skip on Non-Zero Accumulator and Skip on Zero Link REV AND'ed skip conditions.
	CLA	7600	Clear Accumulator Common to all groups.
	LAS	7604	Load Accumulator with Switch Register Logical sequence clears AC then loads it with switch register.
	SZA CLA	7640	Skip on Zero Accumulator then Clear Accumulator
	SNA CLA	7650	Skip on Non-Zero Accumulator then Clear Accumulator. Order of key depression is irrelevant.
	SMA CLA	7700	Skip on Minus Accumulator then Clear Accumulator
	SPA CLA	7710	Skip on Positive Accumulator then Clear Accumulator
	SPA SNA SZL CLA OSR	7774	Skip on Positive Accumulator and Skip on Non-Zero Accumulator and Skip on Zero Link, then clear accumulator and load accumulator with the content of the switch register.

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	CLA	SMA SPA	SZA SNA	SNL SZL	0 1	OSR	HLT	0

LOGICAL SEQUENCES:
 1 (Bit 8 is Zero) — SMA or SZA or SNL
 (Bit 8 is One) — SPA and SNA and SZL
 2 — CLA
 3 — OSR, HLT

GROUP 2 MICROINSTRUCTION FORMAT

FIGURE 2-5

CHAPTER 3
INTERCEPT JR. PROGRAMMING EXAMPLES

INTRODUCTION

The reader who is not familiar with elementary programming techniques, two's complement arithmetic and octal coding, should study Appendix A and the IM6100 brochure for a description of the instruction set before continuing with this section. The MONITOR program will be used to illustrate the use of various techniques. The SETPC and MEM keys may be used to look at the ROM locations shown. The MONITOR listing is in Chapter 8.

EXAMPLE 1 - INCREMENTING MEMORY DATA

```
.  
. .  
6624    2000    AINC,    ISZ SAVPC    /Increment the user PC  
6625    5200                JMP MICRO  
6626    5200                JMP MICRO    /Return for new Micro Command  
. . .
```

This technique uses the ISZ instruction to directly increment memory data without needing to bring it into the AC first. Note the use of the JMP MICRO instruction twice in case the data was 7777 and a skip was performed. NOP instruction after ISZ can also be used to avoid the effect of the skip on the program.

EXAMPLE 2 - DECREMENTING MEMORY DATA

```
.  
. .  
6430    7340    DECPC,    CLA CLL CMA /Set AC to -1  
6431    1000                TAD SAVPC    /Add data in SAVPC  
                        (location 0000)  
6432    3000                DCA SAVPC    /Restore decremented data  
. . .
```

Note the use of the microinstruction combination CLA CLL CMA to clear the AC and the link and then to complement the AC, resulting in 7777 in the AC and 0 in L. By adding the contents of location SAVPC to the AC in two's complement arithmetic, a decrement is effectively performed. Note that the logical sequence of microinstruction execution is chosen for usefulness. It would be of no value to complement the AC first and then to clear it.

EXAMPLE 3 - PROGRAMMING TIME DELAYS

```

.
.
.
6203      3157      DCA      SAV4      /Store wait count in SAV4 and clear AC
6204      1223      TAD      TK1       /Get the Time constant
6205      3144      DCA      TIME      /Place in the timer
6206      2144      ISZ      TIME      /Time out 2.4 ms at 2.46 MHz
6207      5206      JMP.-1     /Jump back one location
.
.
.
6223      7620      TK1,     7620     /-112
.
.
.

```

This sequence is part of SWDB, the switch debounce routine described in Chapter 8. The AC is cleared (incidentally while depositing in SAV4), and the constant TK1 is fetched from the current page address 6223. It is stored in the page 0 location 0144 and ISZ instructions are successively executed until the timer goes to zero and the jump-back instruction is skipped. The delay produced may be calculated by counting the number of major states in each instruction executed and multiplying by the state time. Thus, ISZ requires 16 states and JMP requires 10, so these 26 states are gone through a total of 112 times, for a total of 2912 states. Adding in the states for the DCA, TAD and DCA (11 + 10 + 11 = 32), we have a total of 2944 states. With a 2.46 MHz clock rate, the state time is 813 ns so the delay is (0.813 x 2944) microseconds = 2393.472 microseconds or approximately 2.4 milliseconds.

It is also instructive to note that the location TIME is in page 0, whereas the constant TK1 is stored in the current page (page 31). In this case, RAM happens to be available only in page 0 and 1 and by keeping TIME in page 0, the ISZ instruction in page 31 was able to directly reference the location TIME in page 0. Obviously, ISZ instructions may only reference RAM locations.

EXAMPLE 4A - ADDRESSING MODES

The user should note that a characteristic of page addressing results in the octal coding for two memory reference instructions on different pages being identical when their operands are in the same relative location on the respective pages.

```

    ⋮
0020      5225      /JMP to location 25 on current
                    page, for example to 0025

    ⋮
0220      5225      /JMP to location 25 on current
                    page, for example to 0225

```

The user should enter these two instructions at the two locations specified. By using the SIN, single instruction key, to execute the instruction, the user will see how the addresses are referenced.

Note that memory reference instructions can reference 400g locations directly, 200g on page 0, and 200g on the page containing this instruction. If the instruction happens to be on page 0, then only locations 0 to 177g are directly addressable (see Example 10).

EXAMPLE 4B - ADDRESSING MODES

The user should enter these instructions.

```

    ⋮
0020      5625      /JMP indirect via 0025

    ⋮
0025      0010      /Pointer to 0010g

    ⋮
0220      5625      /JMP indirect via 0225

    ⋮
0225      0010      /Pointer to 0010g

    ⋮

```

Now, by using the single step key at locations 0020 or 0220, the address should change to 0010 showing that an indirect reference has been made.

The pointer (location containing the effective address) can contain a full 12 bits of address, so the program can branch anywhere in the 4K address space by jumping indirect.

When constants and pointer addresses are stored in page 0, references may be made to them from any page, avoiding the necessity of storing them on each page that needs them.

EXAMPLE 5 - INDIRECT ADDRESSING USED IN TABLE MANIPULATION

This example is taken from the SHELL routine described in Chapter 8. It is a common technique of passing program control to one of several possible sequences by adding an index to a base address.

At the point that the following sequence is entered, the accumulator contains an octal number from 0 to 13 which stands for the routines MICRO, BIN, BLK, SIN, RUN, HALT, RESET, SETPC, DECPC, DEP, INSAC and SHELL respectively.

```

:
6402      1207      TAD GOTO      /add base address to constant
6403      3147      DCA POINT      /store pointer in POINT
6404      1547      TAD I POINT     /get routine starting address
6405      3147      DCA POINT      /phase starting address in POINT
6406      5547      JMP I POINT     /go to the routine
6407      6410 GOTO, GOTO +1      /base address
6410      6600      MICRO          )
6411      7622      BIN            )
6412      6474      BLK            )
6413      7400      SIN            )
6414      6436      RUN            )
6415      6434      HALT           )
6416      6165      RESET          )
6417      6543      SETPC          )
6420      6430      DECPC          )
6421      6502      DEP            )
6422      6425      INSAC          )
6423      6400 BUG, SHELL         )
:

```

TABLE OF ROUTINE
STARTING ADDRESSES

Note that location 6407, labeled GOTO contains base address 6410, so by adding a number from 0g to 13g to 6410, a number from 6410 to 6423 is obtained. This number is stored in POINT.

Now, the effective starting address is obtained by executing a TAD indirect through POINT, for example contents of POINT used as operand address. Thus, if AC contained 3g, then 6413 would be stored in POINT, and TAD I POINT would place 7400 in the AC to be again stored in POINT. This time an indirect jump through POINT loads 7400 into the program counter.

Of course, POINT had to be stored in RAM and since pages 0 and 1 are in RAM, POINT was chosen to be in page 0, in order that the upper ROM pages could reference it. It can be seen that indirect addressing makes writing programs easier in mixed RAM-ROM memory where memory references cannot be easily confined to small relative address displacements. See Table 3-1 for a list of pages and their memory locations.

TABLE 3-1

PAGE	MEMORY LOCATIONS
0	0-177
1	200-377
2	400-577
3	600-777
4	1000-1177
5	1200-1377
6	1400-1577
7	1600-1777
10	2000-2177
11	2200-2377
12	2400-2577
13	2600-2777
14	3000-3177
15	3200-3377
16	3400-3577
17	3600-3777
20	4000-4177
21	4200-4377
22	4400-4577
23	4600-4777
24	5000-5177
25	5200-5377
26	5400-5577
27	5600-5777
30	6000-6177
31	6200-6377
32	6400-6577
33	6600-6777
34	7000-7177
35	7200-7377
36	7400-7577
37	7600-7777

EXAMPLE 6 - THE JMS INSTRUCTION AND INDIRECT ADDRESSING

A very important use of indirect addressing is in returning to a main program from a subroutine. Appendix A shows how two programs may be linked using JMP instructions. The JMS instruction's usefulness lies in the fact that only one copy of a subroutine need be stored, for example in page 0, and a program anywhere in main memory may call it. INTERCEPT JR. uses a "last-in-first-out" (LIFO) or "pushdown" stack in page 0 to store subroutine return addresses. This allows nesting of subroutines and calling subroutines stored in the MONITOR ROM by linking through RAM. For further details refer to Appendix L.

Our example will demonstrate the use of the JMS instruction in RAM, and the use of indirect addressing to return.

The user should enter these instructions.

0020	7240	CLA CMA	/AC set to 7777
0021	4100	JMS 0100	/Jump to subroutine starting at 0100
0022	7240	CLA CMA	/AC set to 7777
0023	7402	HLT	
.			
.			
0100	0000		/This location will contain return address
0101	7200	CLA	/AC set to 0000
0102	5500	JMP I 0100	/Return to main program

Single step through this program (by successive depressions of SIN key after initial "CNTRL" "SIN" sequence at program starting address) and the program sequencing will be seen to go from 0020 - 0021 - 0100 - 0101 - 0102 - 0022 - 0023. In between, it will be instructive to look at location 0140 where the AC is saved by the MONITOR. The AC will initially be set to 7777, then the subroutine clears it, and then the main program again sets it to 7777. The JMS instruction stores the return address, namely 0022 in location 0100 so that upon executing the JMP indirect via 0100, the main program can be rejoined in sequence.

If a 1K RAM option card is available, the user could relocate the main program in an upper page and executive the same program provided the subroutine remained in page 0. The subroutine could be moved to a page different from page 0 or the main program's page but then an indirect JMS would have to be executed. We can illustrate this in page 0 as follows:

0020	7240	CLA CMA	/AC = 7777
0021	4424	JMS I 0024	/Jump via pointer in 0024
0022	7240	CLA CMA	/AC = 7777
0023	7402	HLT	
0024	0100		/pointer address
.			
.			
0101	7200	CLA	/AC = 0000
0102	5500	JMP I 0100	/Return

An extra location to store the pointer is needed.

EXAMPLE 7 - AUTOINDEXING

Example 3 showed how a simple loop could be programmed using the ISZ and JMP instructions.

The IM6100 treats memory locations 0010 through 0017, in page 0, in a unique manner. Whenever an instruction makes an indirect reference to any of these locations, the content of the location is incremented before it is used as an operand. These locations can, therefore, be used in indexing applications. The incrementation is done automatically, provided the location was referenced indirectly, without needing ISZ or TAD and IAC instructions, so this feature is known as autoindexing. When these locations are addressed directly, they act as any other location.

Since the autoindex location is incremented before it is used as an operand, it must be set to one less than the first value desired.

0010			/Autoindex location
:			
0200	7200	CLA	/Clear AC to 0000
0201	1212	TAD 0212	/Get # of locations to be cleared
0202	7041	CMA IAC	/2's complement of AC
0203	3212	DCA 0212	/Store in loop counter
0204	1213	TAD 0213	/Get "starting address -1"
0205	3010	DCA 0010	/Store in autoindex location
0206	3410	DCA I 0010	/Clear location pointed to by 0010
0207	2212	ISZ 0212	/Increment loop counter
0210	5206	JMP 0206	/Jump back two places
0211	7402	HLT	/Stop. All locations cleared
0212	0100	CONSTANT	/# of locations to be cleared
0213	0277	START-1	/Starting address (0300) -1

Note that the autoindex location supplies successive memory address pointers until the counter goes to zero and the program halts. The program will clear locations 0300 to 0377.

EXAMPLE 8 - ADDRESS FIELD MODIFICATION

Instructions and program data may be stored in the same memory. Thus, it is possible to treat instructions as data or data as instructions if this would be of any use.

A powerful programming technique involves performing arithmetic on memory reference instructions in order to alter the location being referenced. In this case, the instruction is treated as an operand and incremented, decremented, etc. Logical operations such as masking certain bits may also be useful. Such techniques are useful when manipulating large data tables. Example 5 has shown one technique of manipulating jump address pointers.

Consider the following example:

0200	7300	CLA CLL	/Clear AC and L
0201	1213	TAD 0213	/Get # of data items
0202	7041	CMA IAC	/2's complement of constant
0203	3213	DCA 0213	/Store TALLY
0204	7240	CLA CMA	/AC = 7777
0205	0300	AND 0300	/AND contents of 0300 with AC
0206	7450	SNA	
0207	5215	JMP 0215	
0210	2205	ISZ 0205	/Increment address field
0211	2213	ISZ 0213	/Increment TALLY
0212	5204	JMP 0204	/Jump back to check next item
0213	0100	TALLY	/Constant giving # of items to be checked
0214	0777	MASK	/Used to mask off opcode bits
0215	1205	TAD 0205	/Get instruction referencing zero data item
0216	0214	AND 0214	/Zero opcode bits
0217	3221	DCA 0221	/Store address of zero item
0220	7402	HALT	/Halt
0221			/Address of zero item

This program checks data stored in locations 0300g to 0377g, when it encounters a zero data item in the list, it stores the address of this item in 0221 and stops.

Location 0213 initially contains the number of items stored starting in location 0300. The program replaces this number with its negative by two's complementing it. Successive data items are then read, AND'ing with 7777 in the AC. Note that if the AND leaves a non-zero AC, the AND instruction is incremented, stepping to the next item. A logical operation is done with this instruction to strip off the opcode bits when and if a zero data item is eventually detected. For this purpose, the mask 0777 is stored in 0214.

On powering up most locations will be non-zero, so the user can put a zero anywhere he chooses to check Example 8 operation. This technique of modifying instructions is a dangerous one to use in many situations because programs may be unintentionally changed because of an undiscovered "bug". (Modern concepts of structured programming discourage the use of this technique, but it is included because in some microprocessor applications, it might save memory locations.) For example, in this case, every time the program is rerun, locations 0205 and 0213 must be initialized.

EXAMPLE 9 - USING CONDITIONAL SKIPS

Group 2 microinstructions are primarily conditional skips and may be used to test conditions other than the number of passes that have been made through a loop. That is, the program may be made to loop an indefinite number of times until a specific condition is present in the accumulator or link bit. When two or more skip conditions are microprogrammed into a single instruction, the resulting condition on which the decision will be based is the logical OR of the individual conditions when bit 8 is 0, or, when bit 8 is 1, the decision will be based on the logical AND.

In the last example, the SNA instruction was used to skip on non-zero accumulator. The loop would continue as long as the next instruction was skipped and when the AC became zero, the program would jump out of the loop.

Very often conditional skips are used along with Group 1 operate microinstructions. The Group 1 instructions are used to manipulate the AC and L with shift, rotate, set, clear operations to set up these registers for testing with conditional skip instructions. This is used extensively in the MONITOR program, for example, in the routine called HEX (see listing of MONITOR and Chapter 8).

The following segment of code is in the MONITOR locations 6466-6471.

```

      :
      6466  7640      SZA CLA
      6467  5263      JMP OK2
      6470  7260      CLA CMA CML
      6471  5263      JMP OK2
      :
  
```

This segment shows testing of the AC to see if it is zero or not. If AC is not zero, the program jumps to OK2. AC (0) can be tested with instructions such as SMA, skip if AC is less than 0, SPA, skip if AC is greater than or equal to 0, and their combinations, and the Link can be tested with instructions such as SZL, skip if Link = 0, SNL, skip if Link = 1. Combinations are possible which test these bits in one instruction, for example, SMA, SNL, skip if AC is less than 0 OR if Link = 1, or SPA SZL, skip if AC is greater than or equal to 0 and L = 0.

The user should note that SMA SNL will produce a skip on minus AC OR non-zero link OR both, whereas SPA SZL will produce a skip on plus AC AND zero link (both conditions must be present for a skip).

The example also shows how microprogrammed combinations of microinstructions may be used to set various constants into the AC.

The instruction in location 6470 sets AC to -1 by first clearing it, then complementing it and the Link to get two's complement of -1, (77778).

EXAMPLE 10 - FLOWCHARTING A PROGRAM

Flowcharts may be used to represent hardware operation as well as to represent an algorithm to be implemented in software.

As an example of an algorithm, or computational procedure, we shall work out a program to computer the product of two octal numbers.

PROBLEM: Computer the product of two octal numbers.

ASSUMPTION: The numbers are positive integers and their product does not exceed 4095_{10} or 7777_8 .
The 1st operand is not zero.

SOLUTION: Many different multiplication algorithms exist. We shall choose a simple, inefficient one which is easy to understand and flowchart.

Add one number repeatedly to itself using a second number to determine the number of additions.

The program will make use of a memory reference instruction known as "Increment and Skip on Zero". The ISZ instruction adds a 1 to the referenced data word and then examines the result of the addition. If the result is not zero, the program continues in sequence, performing the instruction following the ISZ. If the result is zero, the instruction following the ISZ is skipped (by incrementing the Program Counter again). In either case, the result of the addition replaces the original data word in memory.

By computing the 2's complement of one operand (data word) and referencing it with the ISZ instruction, we can repeatedly add the second operand to itself until the desired product is obtained. At this point, the counter becomes zero and the loop exit is taken.

After entering the program as shown, data may be entered into locations 0032 and 0033, the Program Counter is set to the starting address, and the program is run.

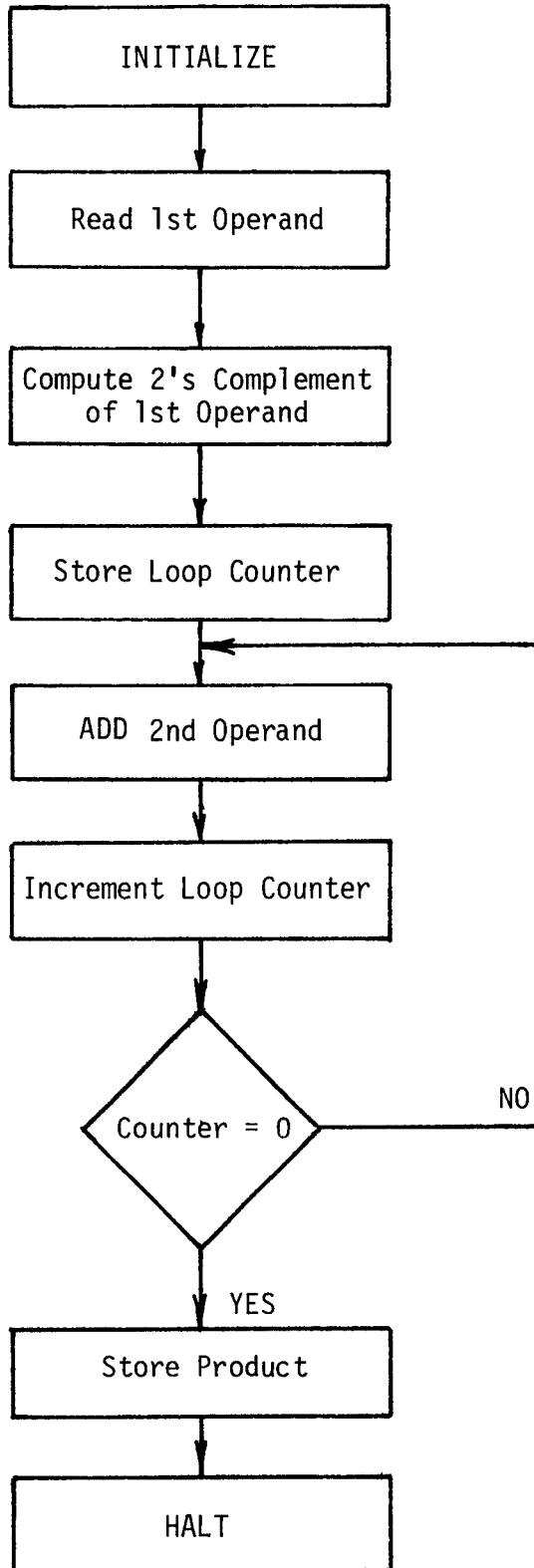


FIGURE 3-1

0020	7300	CLA	CLL
0021	1032	TAD	0032
0022	7041	CMA	IAC
0023	3031	DCA	0031
0024	1033	TAD	0033
0025	2031	ISZ	0031
0026	5024	JMP	0024
0027	3031	DCA	0031
0030	7402	HLT	
0031		loop counter and final product	
0032		1st operand	
0033		2nd operand	

PROGRAM TO MULTIPLY TWO OCTAL NUMBERS TOGETHER

CNTRL	SETPC	0	0	2	0	
CNTRL	MICRO	OPR1		CLA		CLL
CNTRL	TAD	0	0	3		2
CNTRL	OPR1	CMA		IAC		
CNTRL	DCA	0	0	3		1
CNTRL	TAD	0	0	3		3
CNTRL	ISZ	0	0	3		1
CNTRL	JMP	0	0	2		4
CNTRL	DCA	0	0	3		1
CNTRL	OPR2	HALT				
CNTRL						
CNTRL	SETPC	0	0	3		2
CNTRL	MEM	1st OPERAND				
	MEM	2nd OPERAND				
CNTRL	SETPC	0	0	2		0
CNTRL	RUN	Display shows product				

enter program

execute program

For example, if 1st operand is 00004 and 2nd operand is 0010, the display will show 0040. The user will also find it instructive to load small numbers as operands and single-step through the program to verify that the program follows the flowchart. Thus, set the PC to 0020, then press "CNTRL", "SIN" and then press the "SIN" key repeatedly. Each time it is pressed, the program executes one SINGle instruction. At any point, the user may set the PC to 0410 to examine the contents of the accumulator (this is explained further in Chapter 8) and resume execution of single instructions by resetting the PC to the last address the user had stopped at and continuing with SIN key depressions.

The yellow Inspect AC key may be used in the MICRO mode to inspect AC contents at any time. The user may alternately single step and press IAC to note the change in the AC. Note that when the program is fully executed in SIN mode, location 0031 is found to contain the loop counter value 0000 instead of 0040 even though the AC contained 0040 prior to single stepping the DCA 0031 instruction.

The reason is that the MONITOR saved the loop counter and placed a breakpoint in its place and even though the single instruction was executed properly, the loop counter was restored. A complete explanation may be found in Chapter 8 in the description of the SINGLE INSTRUCTION EXECUTE routine.

The DCA 0031 in location 0030 may be replaced by a NOP, 7400 while single stepping. In the RUN mode, of course, the program will halt showing the final product in location 0031. The Inspect AC feature could be left on in the RUN mode, but since the AC is cleared when the DCA is executed, this is not particularly useful.

It is instructive to replace the DCA 3031 in location 0027 with a JMP 0020 or 5020, then running the program with the Inspect AC mode on. The flickering of the display reflects the continually changing contents of the AC as the program is executed repeatedly. Use the RESET switch to get out of this loop.

The user will find it useful to rewrite the program to make the assumptions less restrictive. For example, a check could be included to test for a zero 1st operand and, if the test was true, the product zero could be immediately calculated. Tests for negative operands could be included and/or checks for arithmetic overflow.

EXAMPLE 11 - BIT MANIPULATION

Often, it is necessary to set, clear or determine the status of individual bits in a word. For example, a peripheral interface may be returning the status of various devices, and the processor must take action conditional on the status of these flags.

There are several methods. In one, the AC is rotated until the desired bit is in the link and then group 2 operate micro-instructions are used to skip conditionally on the link status. This technique is illustrated in Example 9. Another method is to AND a mask word with the AC, zeroing out all bits except the one to be tested and then testing the AC for zero.

This technique will be illustrated with an example from the SIN routine in the MONITOR.

```

      :
      7450    1400    INDB,  TAD I SAVPC  /get the instruction
      7451    0262                AND LOT    /mask out indirect bit
      7452    7650                SNA CLA    /test; is bit set
      7453    5564                RETURN     /no; return with true
                                   address in TIME
      7454    1544                TAD I TIME  /yes; get true address
      7455    3144                DCA TIME   /place it in TIME
      7456    5564                RETURN     /return with true address
                                   in TIME
      :
      7462    0400    LOT,    0400        /AND mask word
```

This routine INDB, determines the effective address referenced by an instruction and places it in location TIME. By AND'ing the instruction with 0400, the AC will be non-zero if the indirect bit, bit 3, is set and zero if this bit is zero.

The methods for setting and clearing bits are similar. One can rotate the bit into the link and then use group 1 microinstructions to clear or set the link. This has the advantage that rotates may be combined with link bit operations in one instruction.

To clear a bit, one can AND the word in AC with a word containing one's everywhere except in the desired bit position. To set a bit, one can add a word containing zero's everywhere except in the desired bit position. This technique is used by the bit set routines in the MICROINTERPRETER, ROM locations 7243-7275.

The next example shows the use the MQ register in logical operations. It will be seen that this register may also be used in bit manipulation operations.

EXAMPLE 12 - LOGICAL OPERATIONS

Boolean operations play an important role in computer logic. We have seen examples of how the AND instruction can be used to mask out selected bits.

The NOT or logical complement operation is easily performed by placing the logical data word in the accumulator and executing a CMA, complement AC, instruction.

The inclusive OR operation is performed by placing one logical operand into the MQ register (executing an MQL - load MQ from AC), loading the second logical operand into the AC, then executing an MQA instruction (contents of the MQ are OR'ed with contents of the AC).

Any Boolean operation may be synthesized using combinations of the basic AND, OR and NOT operations.

EXAMPLE 13 - I/O PROGRAMMING

Chapter 7 and Chapter 8 give examples of I/O instructions as used in INTERCEPT JR.

There are three methods by which information may be transferred between INTERCEPT JR. and peripheral devices:

- 1) DMA I/O transfer
- 2) Interrupt I/O transfer
- 3) Programmed I/O transfer

The first method involves Direct Memory Access, DMA, by an I/O device and allows for high speed transfers of blocks of data at essentially the memory cycle rate. The transfer is controlled without processor intervention on a "cycle stealing" basis. That is, the I/O device requests a DMA cycle and the processor grants it at the end of the current instruction. (See Figure 17 of the IM6100 brochure). The processor tri-states its bus drivers and from that point on, as long as the DMA REQ line is active, the device controls the DX bus and data transfers on the bus. Typical DMA using devices are disks, tapes and CRT screen refresh circuits.

INTERCEPT JR. primarily uses the last two methods. Both of these require CPU intervention. Interrupt transfers use the interrupt system to service one or more peripheral devices simultaneously, permitting processing to be performed concurrently with data I/O operations.

Both methods use the AC as a data buffer for transfers in both directions.

Interrupt programming is especially useful in real time systems which are required to respond to real time events. The time spent waiting for a change in device status is greatly reduced or even eliminated. This is done by writing I/O handling routines which are separate from the main program and using the interrupting capability of I/O devices to enter these routines only when the I/O device is either ready to perform a data transfer or requires CPU intervention. Thus, as long as the device does not request an interrupt, the mainline program may continue to run and time is not wasted "polling" I/O devices for changes in status.

In INTERCEPT JR., the control panel timer generates interrupt requests at periodic intervals. The display refresh routine that periodically drives the LED displays is an example of an I/O handling routine. When the main program is interrupted, a method of returning to it after servicing the interrupt request is necessary. INTERCEPT JR. saves the current content of the PC in location 0000g of the memory and fetches the next instruction from location 0001g if an external I/O device requests an interrupt.

In the case of a control panel interrupt, the return address is stored in location 0000g of panel memory. This is the same as 0000g of page 0 of the main memory in the INTERCEPT JR.

For further details on device interrupts and CP interrupts, refer to the IM6100 and IM6101 data sheets.

The third, and slowest method, that of programmed data transfer, is also the simplest, needing a minimum of hardware support. The INTERCEPT JR. PIEART board uses this technique. The processor, upon recognizing an I/O instruction, opcode 6g, places the instruction on the DX bus during $IOT_A \cdot LXMAR$. The selected device communicates with the CPU through four control lines--C0, C1, C2 and SKP. The control line SKP, when low during an IOT, causes the CPU to skip the next sequential instruction.

The INPIE, TALK, LISN, and READ routines of the MONITOR should be studied to see the use of IOT's in programmed data transfer.

For example, the print to TTY routine is as follows:

```

.
.
.
7600    6163    TALK,      SKIP2      /Skip on clear Xmit buffer
7601    5200                JMP.-1     /Xmit buffer not yet clear
7602    6161                WRITE1     /Write AC to Uart Xmit buffer
7603    3144                DCA TIME  /Clear AC and store the
                                old character in TIME
7604    5564                RETURN

```

Note the use of the SKIP2 instruction to implement a "wait" loop. When the condition is satisfied, the loop is exited. The device must activate the SKP line back to the CPU in order for the CPU to skip the next instruction.

The WRITE1 instruction is another IOT used to write the AC to the UART. (See Chapter 7 for device address codes and command codes.) Refer to the IM6100 and IM6101 data sheets for more information.

The next chapter describes dedicated IOT instructions used in INTERCEPT JR. namely 6400 - Load Display, 6402 - Enable/Disable CP Timer, 6403 - IOT CPREQ, 6406 - IOT Reset, 6407 - IOT RUN. The experienced user may use these to shut off the timer and perhaps use subroutines in the MONITOR for his own purposes, for instance, display information other than the USERPC and its contents.

EXAMPLE 14 - TELETYPE I/O USING MONITOR CALLS

The following program makes use of the MONITOR ROM PIE-UART subroutines by calling them via the software stack mechanism.

The control panel interrupt requests must be shut off to prevent timing difficulties.

```

0100    7340    Set AC to 7777
0101    6402    Disable CP request timer
0102    4161    CALL
0103    6340    PIE initialization routine INPIE entry address
0104    4161    CALL READ from
0105    7613    Teletype routine
0106    4161    CALL TALK, the print
0107    7600    to TTY routine
0110    5104    Jump back for next character

```


Note that the stack mechanism requires that the CALL instruction (JMS 0161) be followed by the entry address of the subroutine. (See Appendix L, ROM Based Subroutine Calls)

EXAMPLE 15 - PRINTING UNDER KEYPAD CONTROL

The following program will print ASCII characters on a Teletype under control of the INTERCEPT JR. board.

Refer to Appendix F for the ASCII character set.

070	7340		STA STL	/Disable
071	6402		IOT TIMER	/Control panel timer
072	4161		CALL	/Initialize
073	6340		INPIE	/PIEART interface
074	7300	BACK	CLA CLL	
075	4161		CALL	/Wait for keypad
076	6156		CLKPD	/To clear
077	4161		CALL	/Read octal
100	6441		HEX	/Data from keypad
101	7004		RAL	/Shift three places
102	7006		RTL	/Left and swap bytes
103	7002		BSW	/To determine leading code digit
104	1121		TAD K0002	/MSB of ASCII code always one
105	7500		SMA	/Is 2nd ASCII digit 4,5,6,7?
106	7001		IAC	/No, 1st digit must therefore be 3
107	7002		BSW	/Yes, 1st digit must be 2
110	3122		DCA TEMP1	/Store temporarily
111	4161		CALL	/Wait for clear
112	6156		CLKPD	/Keypad
113	4161		CALL	/Read 2nd octal
114	6441		HEX	/Digit
115	1122		TAD TEMP1	/Assemble ASCII character
116	4161		CALL	/Transmit character
117	7600		TALK	/To printer
120	5104		JMP BACK	/Go back for next character
121	0002	K0002,	0002	
122	0000	TEMP1,	0000	

Appendix F shows that the 8-bit ASCII character codes have the property that if the left octal digit is 2, the second octal digit is 4, 5, 6 or 7, and if the left octal digit is 3, then the second octal digit is 0, 1, 2 or 3.

This program allows the user to enter characters as two successive octal digits.

Note that this assumes the eighth (parity) bit is always set.

EXAMPLE 16 - PROGRAM TO DEMONSTRATE I/O TO 6957 AUDVIS MODULE

0225	7201		CLA IAC	/Set AC=0001
0226	6402		ENDIS TIMER	/Shut off CP timer
0227	7000		NOP	
0230	7000		NOP	
0231	7604	READ,	LAS	/Load keypad to AC
0232	7450		SNA	/Key depressed?
0233	5231		JMP READ	/No, go back to try again
0234	6404		LD DISPLAY	/Display AC on LED register
0235	6401		CLOCK	/Click speaker
0236	5231		JMP READ	

The first two instructions shut off the control panel interrupt timer. The three instruction loop in locations 231, 232, and 233 cause the processor to wait until a key is depressed, and when this occurs, to load the LED register with the AC and CLICK the speaker.

While a key is depressed, the processor executes the instructions

LAS	(15 major states)
SNA	(10 major states)
LD DISPLAY	(17 major states)
CLOCK	(17 major states)
JMP READ	(10 major states)

continuously, and the speaker "clicks" merge into a high pitched beep. The fundamental frequency of this "beep" is easily calculated by counting the number of major states in the above instruction sequence, multiplying by twice the clock period and taking the reciprocal of this number.

In this case, there are 69 major states; and, assuming a 2.56MHz crystal, the clock period is 390 ns, and the "beep" frequency is $1 / (69 \times 2 \times 390 \times 10^{-6}) = 18 \text{ KHz}$.

Now change the instruction in location 0236 to 5230. This adds a NOP, or 10 more major states to the loop, decreasing the frequency of the beep. By placing 5227 in location 0236, the frequency is lowered further. This program enables the user to find out which DX line each key is connected to.

Instead of a beep, the program can be made to click on each key depression by replacing the two NOPs with 4161 and 6156. This calls the CLKPD subroutine which waits for a clear (fully released) keypad before returning to the calling program.

The action of the HEX program which encodes key depressions in order to generate MONITOR program subroutine starting addresses may be easily seen by replacing the three instruction keypad read loop in locations 231, 232 and 233 with the sequence 7000, 4161, and 6441. As before, 4161 is a JMS to the top of the RAM subroutine stack and 6441 is the starting address of the HEX routine. Descriptions of these programs may be found in Chapter 8, and a discussion of the software stack may be found in Appendix L.

The program just entered should have looked like this:

0225	7201
0226	6402
0227	4161
0230	6156
0231	7000
0232	4161
0233	6441
0234	6401
0235	6404
0236	5227

EXAMPLE 17 - REAL-TIME PROGRAMMING

A. MONITOR subroutines

Note that when using MONITOR subroutines via the stack mechanism, the CP timer should in general be disabled. The stack base is initialized only on power-up, but there is always a slight chance that when the user calls a subroutine and the user program is setting up the return linkage, a CP interrupt with the resulting CALLs to REFSH, SWDB, CLKPD could disturb the locations used to set up the user subroutine return.

B. Programming for user-generated interrupts

Programs using input and output routines spend a lot of time in loops (skip on device ready flag instruction followed by JMP*-1) waiting for a peripheral device to accept or transmit data. The processor can spend this time productively by using the interrupt facility to signal external conditions to the running program. These external conditions could be peripheral device flags (ready for operation, operation complete, etc.) or alarm conditions (power fail detected).

When the interrupt system is enabled (via execution of the ION instruction), then whenever a device generates an interrupt request to the running program, the following operations occur:

- 1) The instruction currently in execution is completed.
- 2) The INTGNT (interrupt grant) signal is activated.
- 3) The contents of the PC are stored in location 0000g.
- 4) The interrupt system is turned off so no further interrupt requests will be acknowledged.
- 5) The IM6100 begins executing instructions starting at 0001g.

Location 0001g usually contains a direct or indirect JMP to the entry address of an interrupt service routine. In simple systems, the interrupt handler may begin at 0001g.

The interrupt handler in general must perform the following:

- 1) It must save processor status. In general, this means the contents of the AC, L, MQ, instruction and data field registers, and any other data required for proper resumption of execution. The structure of the mainline program (background program) and the interrupt handlers (foreground programs) determines the amount of information needed to be saved.
- 2) The various I/O devices must be polled to determine which one generated the interrupt. Upon identification, control must be transferred to the proper device service routine.
- 3) The required service is performed, and the device interrupt flag is cleared.
- 4) The processor status is restored, the interrupt system is enabled by executing an ION or RTF instruction. Both these instructions take effect (turn the interrupt system on) only after the next sequential instruction, a JMP I 0000g.
- 5) The JMP I 0000g causes execution to resume as if no interrupt had occurred as long as all the required status was saved and restored, and the time delay to the mainline program was not significant.

If a second interrupt occurs while an interrupt is being serviced, the return address in location 0000 would be lost unless the interrupt system is disabled while servicing the first interrupt.

Such a situation may occur when high and low speed devices are being concurrently serviced. To ensure rapid response to the high speed device, the interrupt system is re-enabled before the low speed device has been completely serviced. The interrupt handler must, therefore, save the return address and the low speed device service routine must return indirect through the save address rather than 0000g.

In the INTERCEPT JR., control panel interrupt requests have higher priority than device interrupt requests so a similar situation arises.

Device identification may be accomplished in several ways. Each device must recognize certain IOT instructions addressing it. At least one of these, in a device capable of requesting interrupts, is a "skip on interrupt request" instruction. When this instruction is executed, if the addressed device is grounding the INTREQ line it will also ground the skip (SKP) line. This causes the next instruction (typically an unconditional skip) to be skipped and a JMP instruction executed to the proper service routine.

Typical code follows:

EXAMPLE 17A

HANDLER,	DCA ACSAVE	/SAVE AC
	RAR	/GET LINK,
	DCA LKSAVE	/AND SAVE
	KSF	/KEYBOARD STATUS FLAG?
	SKP	/NO; CHECK PRINTER
	JMP KBD	/YES; GO TO KEYBOARD
		/SERVICE ROUTINE
	TSF	/SKIP ON PRINTER INTERRUPT
	SKP	/NO; GO TO EXIT SEQUENCE
	JMP PRT	/YES; GO TO PRINTER SERVICE
		/ROUTINE
	CAF	/CLEAR ALL DEVICE FLAGS
	JMP EXIT	/AND RETURN
	⋮	
	⋮	
EXIT,	CLA	/CLEAR AC
	TAD LKSAVE	/READ LINK STATE
	CLL RAL	/AND RESTORE
	TAD ACSAVE	/RESTORE AC
	ION	/ENABLE INTERRUPTS AFTER
		/NEXT INSTRUCTION
	JMP I Ø	/RETURN TO MAINLINE PROGRAM

The instruction sequence which determines the interrupting source is called a "skip chain" because of the number of skip instructions.

Skip chains must be designed so that high-speed devices are tested near the top of the chain and that information loss does not occur due to timing problems.

If two interrupts occur simultaneously, the high speed device, being higher up in the chain, will be serviced first, and the low speed device will be serviced as soon as the interrupt system has been re-enabled and the background program has been resumed because it will request another interrupt.

Alternatively, the skip chain may use JMS instructions to the device service routines. Upon termination of the higher priority device service routine, the skip chain is reentered, without re-enabling the interrupt system. Polling of lower priority devices may thus continue and the skip chain must terminate with an ION, JMP I Ø to return control to the mainline program if no further interrupt requests are pending.

Another case arises when during the execution of a low speed device service routine a high speed device requires service. Because the interrupt system is disabled, the request for service may be ignored long enough for information to be lost.

Sometimes, a device may not be capable of high speed data transfers, but it has high priority nevertheless. This is the case of control panel interrupt requests in IM6100 systems such as the INTERCEPT JR. A priority interrupt system can be established through software by the following sequence of operations:

- 1) Begin low priority device service routine by saving all required processor status as well as background program return address in 0000g.
- 2) Execute an ION instruction.
- 3) Clear low priority device interrupt request flag. Interrupt system is now enabled.
- 4) Service the device as required. A high priority interrupt is permissible now without losing the background program linkage.
- 5) Terminate the service routine by restoring processor status and return to the background by an indirect jump via the stored return address.

The INT pushbutton on the AUDVIS card generates an interrupt to the IM6100. The following example illustrates interrupt programming techniques. On receiving the interrupt, the IM6100 automatically saves the PC in location 0000g and executes the instruction in location 0001g. The example interrupt service routine will simply display the current value of AC, re-enable interrupts, and return to the main program.

INTerrupt GrANT (INTGNT) becomes active after an INTREQ is recognized, and is reset after the first IOT instruction is executed. During the time INTGNT is active, CPREQs are gated off by the hardware.

EXAMPLE 17B

```

                                /INTERRUPT SERVICE ROUTINE
0001      3020                  DCA ACSAV      /SAVE AC
0002      1000                  TAD 0000      /GET SAVED RETURN ADDR
0003      3021                  DCA PCSAV      /AND SAVE IN PCSAV
0004      1020                  TAD ACSAV      /RESTORE AC
0005      6404                  DISP          /AC TO DISPLAY
0006      6001                  ION           /RE-ENABLE INTERRUPTS
0007      5421                  JMP I PCSAV    /RETURN TO MAIN PROGRAM

0020      0000      ACSAV,      0000          /AC SAVE LOC
0021      0000      PCSAV,      0000          /PC SAVE LOC

                                /MAIN PROGRAM
                                /INCREMENT AC REPEATEDLY. WHEN INT
                                /PUSHBUTTON IS PRESSED THE INTERRUPT
                                /SERVICE ROUTINE WILL DISPLAY THE
                                /CURRENT VALUE OF THE AC.

0022      6001      START,      ION           /ENABLE INTERRUPTS
0023      7001      LOOP,       IAC          /INCREMENT AC
0024      5023                  JMP LOOP     /AGAIN AND AGAIN

```

In the INTERCEPT JR., the CP timer need not be turned off for user generated interrupts provided that the CP TIMER routine execution time does not interfere with the device interrupt response or service time. This is because the hardware uses INTGNT to gate CP interrupt requests.

However, INTGNT is reset by the execution of any IOT and this would allow CPREQs to get through once again.

Note that the execution of an IOT after an INTGNT is also used by peripheral devices to place an interrupt vector on the DX bus.

Interrupt vectoring is a procedure by which an interrupting device can identify itself eliminating the need for a skip chain. The device places an address (the interrupt vector) onto the DX bus which is used by the processor to branch to the appropriate device service routine. Prioritization of the devices is accomplished in the hardware by a "priority chain" such that a device may request an interrupt only when no higher priority device is also requesting an interrupt.

A user interrupt routine in an INTERCEPT JR. system with vectored interrupt should be functionally identical to one of the following routines:

EXAMPLE 17C-Interrupt Service Routine without timer off:

0000	0000	0000	Return address
0001	3006	DCA AC	Save AC in 0006
0002	1000	TAD 0000	Get interrupt return address
0003	3007	DCA PC	Save return address in 0007
0004	6002	IOF	Vector to user service routine
0005	0000	0000	
0006	0000	AC, 0000	
0007	00	PC, 0000	

Note that the CP Timer is gated off while the instructions in 0001-0004 are being executed by JR. hardware (gate D4).

Also note that the user interrupt service routine should return indirectly through location 0007 (JMP I 0007-5407) and the service routine should use contents of location 0006 as AC.

The user interrupt service routine is quite likely to be interrupted by the CP Timer but the timer routine will return properly to the user routine.

EXAMPLE 17D-Interrupt Service Routine with timer off:

0000	0000	0000	Return address
0001	3006	DCA AC	Save AC in 0006
0002	1000	TAD 0000	Get return address
0003	3007	DCA PC	Save in 0007
0004	7001	IAC	IAC
0005	6402	EN/DIS Timer	Disable timer and vector
0006	0000	AC, 0000	
0007	0000	PC, 0000	

This routine may let one CP Timer request through since the timer oscillator may have already clocked the request FF (D5).

Unfortunately, the only way to guarantee that no timer request interferes with the interrupt service routine execution time is to turn it off in the main program itself. One must do this, in any case, if the interrupt response time is critical.

Please note that locations 0010-0017 are autoindexed and hence they must not be used to save PC since the contents of 0010-0017 will be incremented by 1 before being used if they are referenced indirectly, for example, by an instruction `JMP I 0010 (5410)`.

Note the examples 17C and 17D use the techniques described earlier for a system with a high priority (CP) and low priority (user) device. The IOT instructions reset INTGNT, allowing CP requests to get through and vectoring to the user service routine. In example 17C, the IOF is used purely to vector, as the user interrupt system is already automatically disabled. In more complex priority interrupt systems, interrupt processing for a given device can be interrupted in order to service higher priority devices, and this procedure is facilitated by saving interrupt return addresses and interrupt processor state on a stack similar to the MONITOR subroutine stack.

Real time systems are much harder to debug because of the asynchronous nature of the signals and events. Failures that occur non-repetitively and seemingly at random are very hard to pinpoint. The user must be much more careful in writing and documenting software, and analyze interaction between program segments thoroughly.

As an example of a simple failure because of an asynchronous event, consider an interrupt service routine that did not save location 0000₈.

EXAMPLE 17E

```

      ⋮
0050   ION           /ENABLE DEVICE INTERRUPTS
0051   JMP I Ø      /RETURN TO BACKGROUND
```

Assume that a CP Timer request was generated immediately after the execution of the ION in 0050. Now location 0000 will have 0051, the return address. After the timer service routine, the program returns to 0051 which specifies a `JMP I Ø`, that is, to jump to itself (since 0000 has 0051). So the program will get stuck here and will never get out.

C. SKP Programming

Often the programmer wishes to test the condition of an external device and execute different program segments depending on the result. One way of accomplishing this is to read the device status (with an IOT instruction) into the AC and then use a conditional skip operate instruction to perform the test. Another method uses a single IOT instruction (called a SKIP IOT) which tests on external device and skips the next sequential instruction if the test was successful. The SKP pushbutton on the AUDVIS module is a "device" which may be tested in this manner using a 6405 SKIP IOT instruction. The 6405 IOT in this case also reads the switch register into the AC, but it is possible to have a SKIP IOT which does not modify the AC. In the following example, the switches are read into the AC, and the AC is two's complemented if the SKP pushbutton is not pressed. Finally the AC is displayed on the LED readouts.

EXAMPLE 17F

0020	6405	START,	RDSWRG	/READ SWITCHES AND SKIP IF SKP BUTTON PRESSED
0021	7041		CIA	/NEGATE AC
0022	6404		DISP	/DISPLAY AC
0023	5020		JMP START	

CHAPTER 4

INTERCEPT JR. MODULE

INTRODUCTION

As shown on the schematic, all memory and I/O devices are connected to the IM6100 DX bus. The twelve (12) bit bus carries time-multiplexed addresses and data from memory and I/O devices.

Timing information must be provided to strobe data on and off the bus and select lines are needed to enable the proper devices.

The MONITOR ROM and 256 x 12 RAM are mapped in upper and lower areas of the 4K address space, and it is necessary to select the proper devices during memory I/O.

The keyboard commands must be interpreted after making sure switch bounce does not cause erroneous operation.

The ADDRESS and MEMORY display digits are multiplexed in order to reduce the number of decoder/drivers required.

The IM6100 microprocessor used in the INTERCEPT JR. is the commercial temperature range device and a 2.46 MHz crystal is used in order to ensure operation of the system as battery voltage falls from 6 V to 4.5 V.

TYING ON TO THE DX BUS

The DX bus carries addresses and data at different times. All peripherals and memory address inputs, peripherals and memory data inputs and outputs are connected to the bus. All elements connected to the bus are, therefore, tri-state devices.

Data strobes and device signals must be generated in order to demultiplex data from the bus or multiplex data onto the bus.

The MONITOR ROM, a 1024 x 12 device is mask-programmed at the factory to decode the lower ten (10) bits as an address, and the upper two (2) bits as a chip enable. For example, the MONITOR ROM, as supplied by the factory, has the upper two bits mask programmed to 11 to select the ROM for 6000 to 7777.

When data is read out, the chip puts its data out onto the DX bus. Thus the DX pins on the 6312 are bidirectional (addresses in and data out).

The RAM is 256 x 12, implemented in CMOS by 3 6561 chips, each 256 x 4.

The A₀-A₇ address inputs and the I/O data pins are connected to the DX bus.

ADDRESS DEMULTIPLEXING

Both the ROM chips and the RAM chips have internal address latches. These latches are loaded from the address inputs when the strobe input STR is driven low. When STR is low, the latches are not affected.

When the processor places memory address data on the bus, it drives the signal LXMAR at pin 10 low. This signal, Load External Memory Address Register, is intended to strobe the memory address latches. Note that the chip does not have to be selected in order to latch address information.

DATA DEMULTIPLEXING

After the CPU places a memory address on the bus, a data transfer must take place either into the CPU from memory or from the CPU to memory. The direction is indicated by the XTC line. The various SELECT lines are activated during the data-in and data-out phases of the memory cycle. XTC is high for the first half of a memory cycle (when memory read operations may be performed) and low for the second half (when memory may be written into). Thus XTC may be directly connected to OEH, Output Enable Active High, of the ROM chips and R/W, Write Enable Active Low, of the RAM chips to enable these chips for reading or writing. During XTC high, of course, the RAM may be selected for reading. The memory outputs will not be activated unless the chip has been selected as well as had its output enabled. Otherwise, many chips would be activated at the same time.

Obviously, it would be undesirable to simultaneously read from several devices onto the same DX lines at once.

For this reason, the active low chip select pins on the RAM chips and OEL, Output Enable Active Low, on the IM6312's are connected to the SEL line. This line may be strapped to either the "MEM SEL" line or the AND'ed combination of "MEM SEL" and "CP SEL". These are active low signals generated by the CPU to select user memory, MEM SEL, or control panel memory, CP SEL. With only the Intersil provided control panel ROM in the system, the jumpers should provide the combination AND signal. This combination signal will select memory when either MEM SEL or CP SEL goes low.

Another aspect to be considered is how addressable memory space is partitioned. In the INTERCEPT JR., the MONITOR ROM occupies the highest 1K of the basic 4K address space and the RAM occupies the lowest 256 words of this space. It is possible to program 256 word pages of the 4K address space for RAM into the IM6312 ROM such that it will generate an RSEL, RAM SELECT, signal by decoding the high

order four bits of the address. These fields must obviously be aligned with page boundaries. RSEL is connected to CS₁ of the IM6561's. In the IM6312-002 MONITOR ROM, RSEL is activated by "0000" on DX0, DX1, DX2 and DX3.

RSEL allows random mapping of double page RAM fields within the 4K address space. Note that the base page, or at least the first 16 locations must be writable in order for autoincrement instructions and interrupt instructions to work. Also note that the highest location (7777) should normally be in ROM as it is used as a pointer to power up initialization routines. See Figure 8-1 for a memory map.

Normally the RAM area does not overlap with the ROM area, therefore, one of the RAM chip select pins is kept permanently low by a jumper to GND so that selection depends only on the chip select connected to the SEL line. BVCC is always present for data retention.

The mapping of RAM into ROM space is of significance should the user generate a ROM to be placed in the spare socket which requires this feature. In such a case, the RAM chip select jumper must be connected to the appropriate RSEL pin. The ROM is mask programmed to generate RSEL appropriately.

Please refer to the IM6312 data sheet for further details.

KEYBOARD INPUT

The INTERCEPT JR. uses a 12 switch keyboard which is an ideal situation as there are 12 DX lines. Each key is connected through a 3-state inverting buffer to the corresponding DX line.

When the CPU executes an OSR instruction, OR Switch Register with accumulator contents, it activates the SW SEL, Switch Select, line and OR's the DX bus with the accumulator. SW SEL is used to enable the keyboard buffers thereby giving the means to read the keyboard.

Naturally, it must not respond to illegal key closures (illegal combinations, bouncing, or too many keys being depressed, etc.). These conditions are checked by the firmware, to be described later.

To improve noise immunity, the inputs to the buffers are pulled up to VCC via 1K resistors in a DIP package.

DIGITAL DISPLAY OUT

The INTERCEPT JR. has two display registers, each with four decimal (BCD) digits.

Each register is driven by a type 4511 CMOS BCD-TO-7 segment latch/decoder/driver and four transistors that enable successive digits in turn (H2, J2, Q1, Q3, Q4)*.

The CPU loads the BCD latch with a digit each, and the 34042 quad CMOS latch (G2) with a single bit and this enables two particular digits to display the decoded contents of the BCD latches. In the next cycle, the BCD latches get loaded with the contents of the two adjacent digits and the bit shifts one position in the quad latch, enabling the next digits, and so on. The CPU can blank the displays under keyboard control in order to conserve battery power.

The data in the AC is loaded into the display latches by 'LOAD DISPLAY' at $IOTA \cdot \overline{XTC} \cdot \overline{DEVSEL}$. The 'LOAD DISPLAY' command is generated by IOT decoding circuitry to be described in the next section.

The 2N2222 transistors, when turned on by the shifting bit, connect the LED common cathode to a low voltage. The drivers source current to individual segments, lighting these up for the time that the bit keeps that digit selected (nominally 8 ms at 4 MHz).

IOT PROCESSING

The INTERCEPT JR. uses Programmed Data Transfer techniques for all I/O operations. This technique uses the IM6100 IOT instructions, which have an octal opcode of 6, to initiate peripheral I/O operations. These operations could be sensing of peripheral device status flags, for example, 'is TTY ready', or controlling device operation, for example, "move disk head to next track", or a data transfer operation, for example, "read character". The nature of the operation depends entirely on the device interface circuitry.

The IM6100 also has the capability for INTERRUPT data transfers and DMA data transfer, but these are unused in the INTERCEPT JR. except for console interrupts described in the next section.

When the IM6100 fetches an IOT instruction, it executes an IOTA cycle, during which the entire IOT instruction is placed on the DX bus during LXMAR time. This means external address registers, such as the ones on board memory chips, will all be loaded with the IOT instruction. In order not to have a memory chip respond falsely, the CPU suppresses the MEM SEL signal, and activates the DEV SEL, Device Select, signal. The device address and control information present in bits 3-11 of the IOT instruction are decoded and the DEV SEL signal is used by the peripheral to enable the selected functions.

* These designations are used to identify the devices on the schematic and on the assembled board.

The 340175 CMOS quad latch (B3) is strobed by $\overline{\text{LXMAR}}$ to latch DX3 and DX9, DX10, DX11 from the bus. The 74C42 CMOS BCD to decimal decoder (B4) is fed with AX11, AX10, AX9 and AX3. The AX3 line acts as an enable to the decoder and must be high in order for the D input to the decoder, which is the most significant bit, to be low.

This means that all device addresses in this system should be of the form 1XXXXX. The 74C42 is a control decoder and only eight of its outputs, corresponding to the possible permutations of the three bit control field in the IOT instruction, may be used. Of these eight, only five, corresponding to IOT's with DX3 high and 0, 2, 3, 6 and 7g in their control field, are used. For simplicity we shall assume a device address of 100000 or 40g.

These IOT instructions will now be described:

LOAD DISPLAY, or 6400 is gated along with XTC and DEVSEL through an OR, the 34025 NOR (A3) followed by the 34069 inverter (C4), into the Load Enable pins of the display drivers. During $\text{IOTA} \cdot \overline{\text{XTC}} \cdot \text{DEVSEL}$ time, this control function will load the latches in the display drivers (H2, J2) and the 34042 quad latch (G2) which drives the multiplexing transistors.

IOT RESET, or 6406 is gated along with DEVSEL through the two NOR's (C5) to generate an active low RESET. RESET is also generated on power-up, when the one input of the 34001 NOR gate (C5) is pulled high by the charging .47 microfarad capacitor. The RESET line driven low will clear the IM6100 accumulator, load 7777g into the program counter, and halt the CPU, besides resetting external logic. RESET is activated on power-up through the RC circuit, at any time by pressing the RESET switch or under program control. The RESET line into the IM6100 is sampled at T1 time of the last cycle of an instruction, and the worst case response time is 14 μsec at 4 MHz. The IOT RESET is a software simulation of the direct RESET line needing approximately a dozen instructions. Including the time needed to debounce the keypad, executing the routine, etc., the response time is many milliseconds. Thus the CPU does not actually do a RESET; it is made to clear all registers initialize the PC to 0200 and is then halted.

IOT RUN, or 6407 from the control decoder is gated along with DEVSEL. When enabled by XTC, the RUN/HLT line is driven by a negative going pulse. Each such pulse causes the CPU to alternatively run and halt by changing the state of the internal RUN/HLT flip flop.

IOT CPREQ, or 6403 is gated with DEVSEL through the 34025 NOR (F3) and 34069 inverter (C4) into the active low direct set input of the DFF 74C74 (D5). During IOTA time, DEVSEL will set the DFF and provided that INTGNT is not active and holding off the 34011 NAND (D4), a CPREQ will be issued. The 74C74 is reset by CPSEL.

CP TIMER EN/DIS, or 6402 is an IOT instruction that is used to turn the control panel interrupt timer on or off under program control. The CP timer circuit is formed by two gates (34001 NOR at C5 and 34011 inverter at D4) and an RC circuit (6.8 K R³ and .47 microfarad C⁸) and as long as pin 5 of the NOR at C5 is low, the oscillator is enabled, running and clocking the DFF at D⁵ at a 30 Hz rate. Thus, CP REQuests are issued at a 30 Hz rate (the DFF being reset by CPSEL in between). When IOT instruction 6402 is executed, during IOTA · DEVSEL · XTC time, clock input pin 11 of the 74C74 DFF at C3 is driven low and the rising edge of DEVSEL clocks in the data on DX11 into the flip flop. At this time, the IM6100 is driving the DX bus with the accumulator so if AC11 is high, the DFF is set, and if AC11 is low, the DFF is cleared. If the DFF is set, the CP timer is disabled by holding pin 4 of the NOR gate at C5 at a low. If the DFF is cleared, this gate is allowed to toggle and the timer runs. Note that during normal operation, the CP timer is running, and CPREQ and CPSEL are being generated.

The reason that CPREQ is not activated unless INTGNT is inactive is that control panel interrupt requests have higher priority than device interrupt requests or even DMA requests. Since INTERCEPT JR. uses main memory for both control panel as well as user routines, interrupt return addresses are saved in location 0000g. Thus, if CPREQ were allowed to be active at all times, the user's device interrupt return address could be destroyed by a CPREQ. INTGNT is activated only by INTREQ and is reset by executing the first IOT instruction in the interrupt service routine. At this time, the CPREQ is allowed to get through, as long as the IOT did not disable the CP timer. If the user is implementing an interrupting device interface with PIE interrupts enabled, a single IOT would be used to reset INTGNT, disable CPREQ and get an interrupt vector from the PIE. At the conclusion of the service routine, CPREQ would be re-enabled under program control.

The monitor firmware will be more fully discussed in Chapter 8. For a more detailed discussion of the control panel capabilities of the IM6100, refer to the IM6100 brochure. INTERCEPT JR. uses the same memory address space for control panel, monitor functions and user memory. See the discussion on the monitor program for further details.

OPTIONS

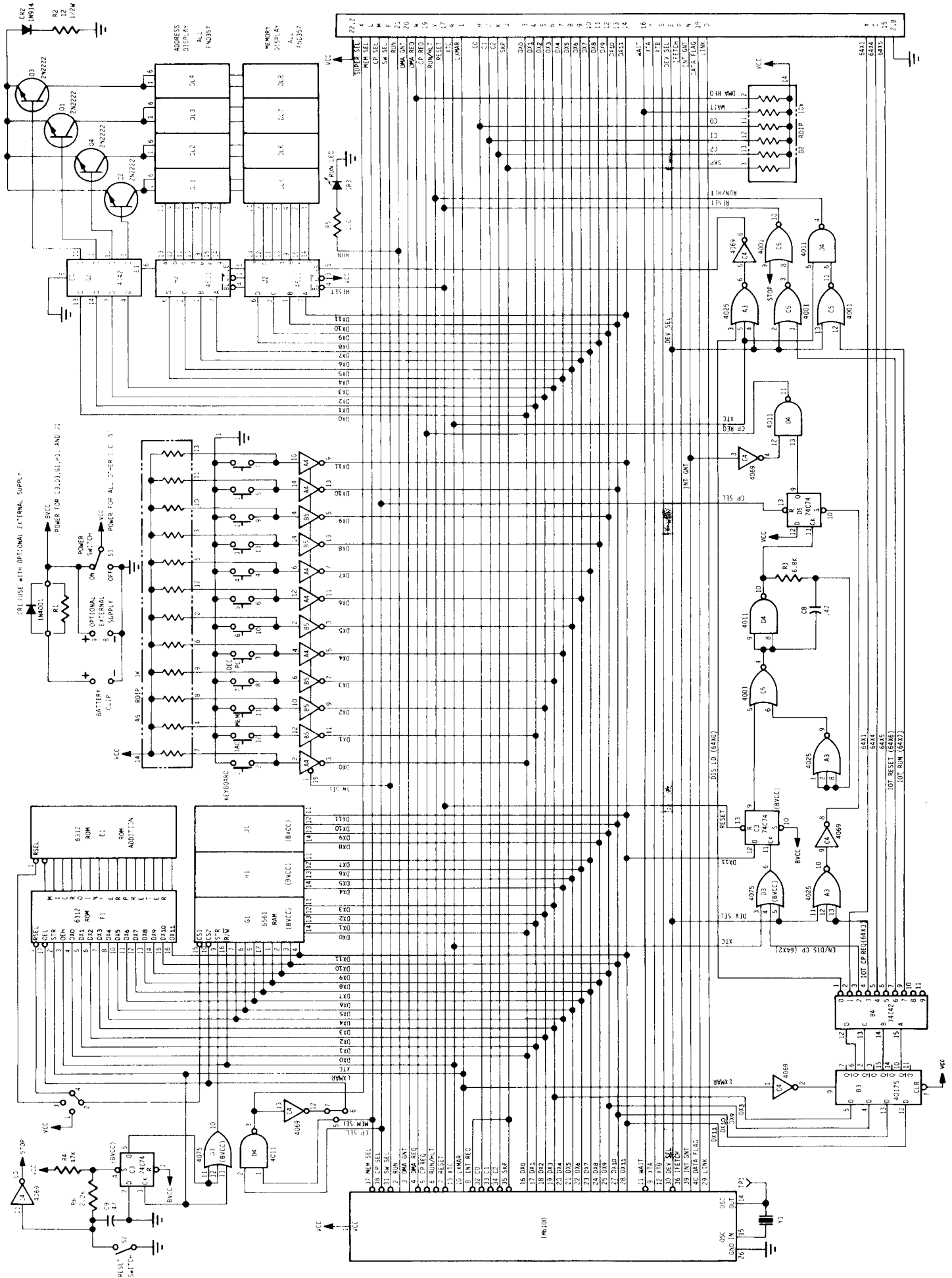
The user may put another IM6312 ROM in the second socket provided on the INTERCEPT JR. board. Extra decoders are not required. The second ROM could contain user and/or factory generated programs such as floating point math routines, I/O handlers, diagnostics programs, utilities, etc. See Appendix K.

As part of the initialization sequence, the MONITOR will also check for the presence of a ROM in the expansion socket. Proper interfacing to the MONITOR requires that any ROM in this socket should be programmed to occupy the 4000g-5777g address area, should have 0764g (two's complement of 7014g) in location 5777g and should have a valid entry point at 4000g.

The following chapters will describe the optional boards that may be plugged into the 6950-INTERCEPT JR. to expand its capabilities. The three connectors on the 6950 board are in parallel and bring out the DX bus, IM6100 control lines, select lines, power connections and unused IOT control lines from the 74C42 decoder (B4).

The basic 256 words of RAM may be disabled by tying chip select high through the jumper option pins provided. This is done by cutting the printed trace between pins 2 and 4 (above MONITOR ROM) and strapping pins 1 and 2 together. This is done when the 6951-M1KX12 JR. RAM MODULE board is to be mapped into the lower 1K field in 000g to 1777g.

The information in this manual and in the IM6100 Family brochure should help the user to design his own I/O interface boards if required.



INTERCEPT JR. MODULE SCHEMATIC

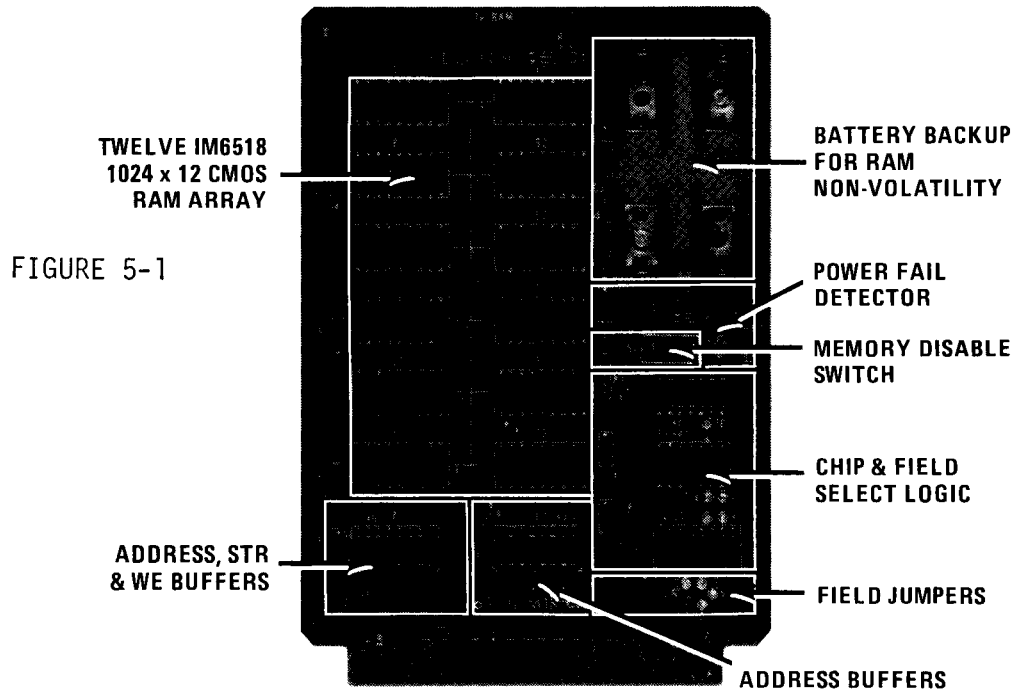
6950-INTERCEPT JUNIOR

REVISION F

CHAPTER 5
JR. RAM MODULE

INTRODUCTION

The JR. RAM MODULE, 6951-M1KX12, pictured in Figure 5-1, allows the user to expand the complexity and size of the programs that may be written up to the 4K word memory size limit.



The board is fully nonvolatile using penlite cells to retain the RAM chips in the low power data retention mode. Thus, the user may write programs on a board, unplug it and use a different board without losing programs. The board may be mapped into memory space according to several jumper options. The board may also be configured as either an Instruction Field or a Data Field by jumper option. (Refer to the IM6100 brochure.)

DISCUSSION

Twelve (12) IM6518 CMOS RAM chips are used to implement the 1024 x 12 array for this board. The IM6518 is organized as 1024 x 1 with separate data-in and data-out pins and ten (10) address pins. (Refer to the IM6508/18 data sheet for further information.) INTERCEPT JR. uses a single bus for all address and data I/O, therefore, the DI and DO pins on the RAM chips are both connected to the respective DX line. The ten (10) address lines are buffered using ten gates from

two 34050 hex CMOS buffers (G1 and G2). One gate is used to buffer XTC. This signal and LXMAR, as previously explained in the discussion of the 6950 board, strobe memory addresses into the RAM chips and enable the chip for data write operations.

The $\overline{\text{SUP SEL}}$ signal is also buffered. This signal selects the RAM for both control panel and main memory use.

A NAND latch is formed by two gates of 34011 quad two input NAND (E3) and can be used to disable memory by grounding one input to one of these gates. Switch S1, DISABLE, is provided for this purpose.

The two most significant bits of address are latched in the 340175 quad D-type latch (G3). This latch provides both true and complemented outputs, and, by connecting the appropriate jumpers to the 34075 three input OR (F3), the 1K RAM field provided by the board may be mapped into any of the four 1K fields of the total 4K memory space addressable by the IM6100 microprocessor.

Since the highest 1K field is occupied by the MONITOR ROM and 256 words of RAM are provided in the lower 1K field by the 6950 module, normally the jumper should be placed to map the RAM into one of the middle 1K areas, for example 2000g-3777g or 4000g-5777g.

If these two fields are being allocated for the PROM board, 6952, the RAM may be mapped into the 1K base field in which it will overlay the 256 words provided in the 6950 board. This will provide the additional 768 words that would otherwise be unobtainable.

Table 5-1 provides the jumper connections for different mappings.

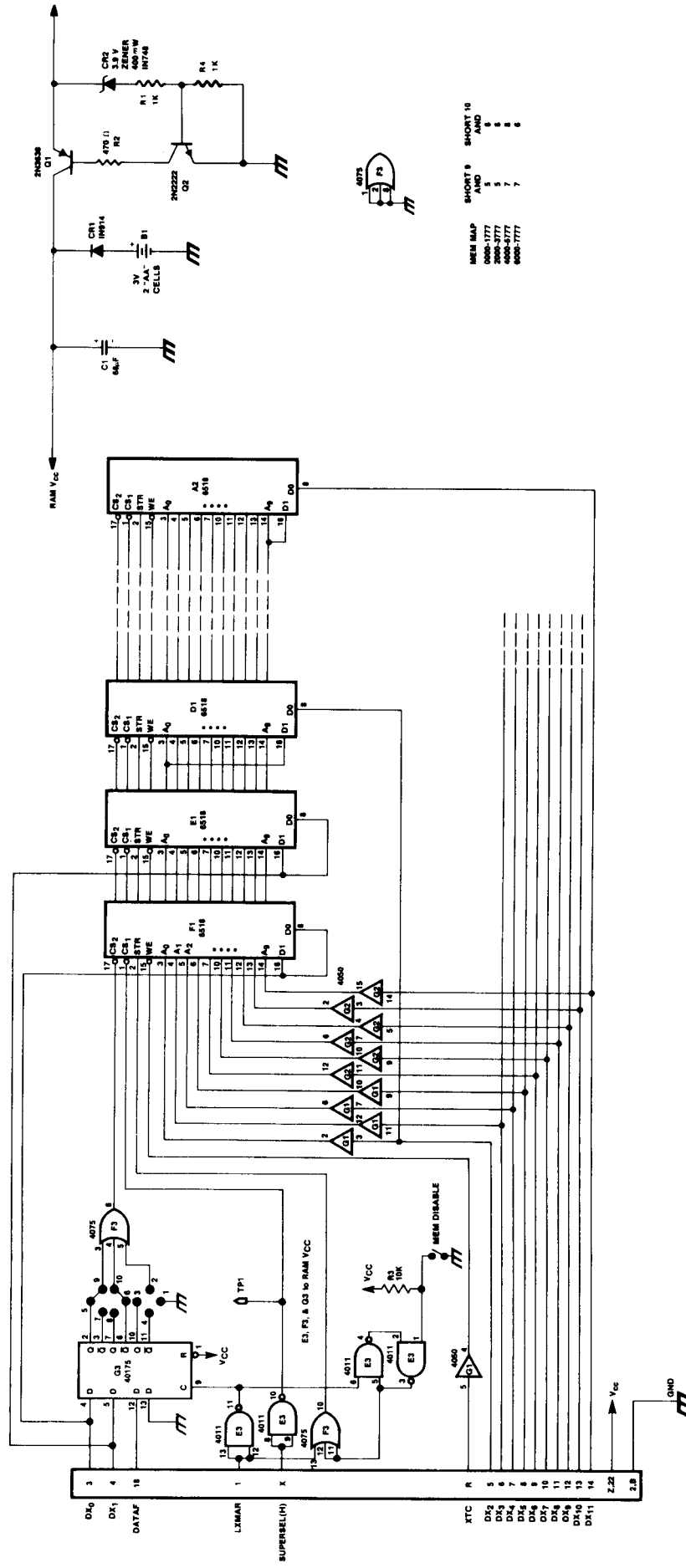
TABLE 5-1

<u>Desired Mapping</u>	<u>Strap* Pins 9, 10</u>
0-1777	To Pins 5 & 8
2000-3777	To Pins 5 & 6
4000-5777	To Pins 7 & 8
6000-7777	To Pins 7 & 6

* These strapping option pins are numbered and located between the 340175 at G3 and the connector pins. For mapping 2000-3777, pins 9 and 5 and pins 10 and 6 are strapped together. Other mappings require cutting the printed trace before adding the new straps.

The board may also be configured to be either an instruction field or a data field by an appropriate jumper connected to the DATAF pin. Normally, the field jumper from test point 2 is connected to V_{CC} and distinctions are not made between IF and DF. These distinctions are usually required only in extended memory systems (Refer to IM6102 data sheet).

The RAM on this board may be made nonvolatile by using two "AA" type penlite cells in the clips provided. If V_{CC} from the "D" cells falls below 3.9 volts, the zener diode CR2 turns off, turning off transistor Q2, which in turn cuts off the series transistor Q1. Diode CR1 becomes forward biased, and the "AA" cells power the RAM array in the data retention mode.



MEM MAP	SHORT 9	SHORT 10
0000-1777	AND	AND
1778-3555	5	6
3556-5333	7	8
5334-7111	7	8
7112-8889	7	8

REV D

6951-M1KX12

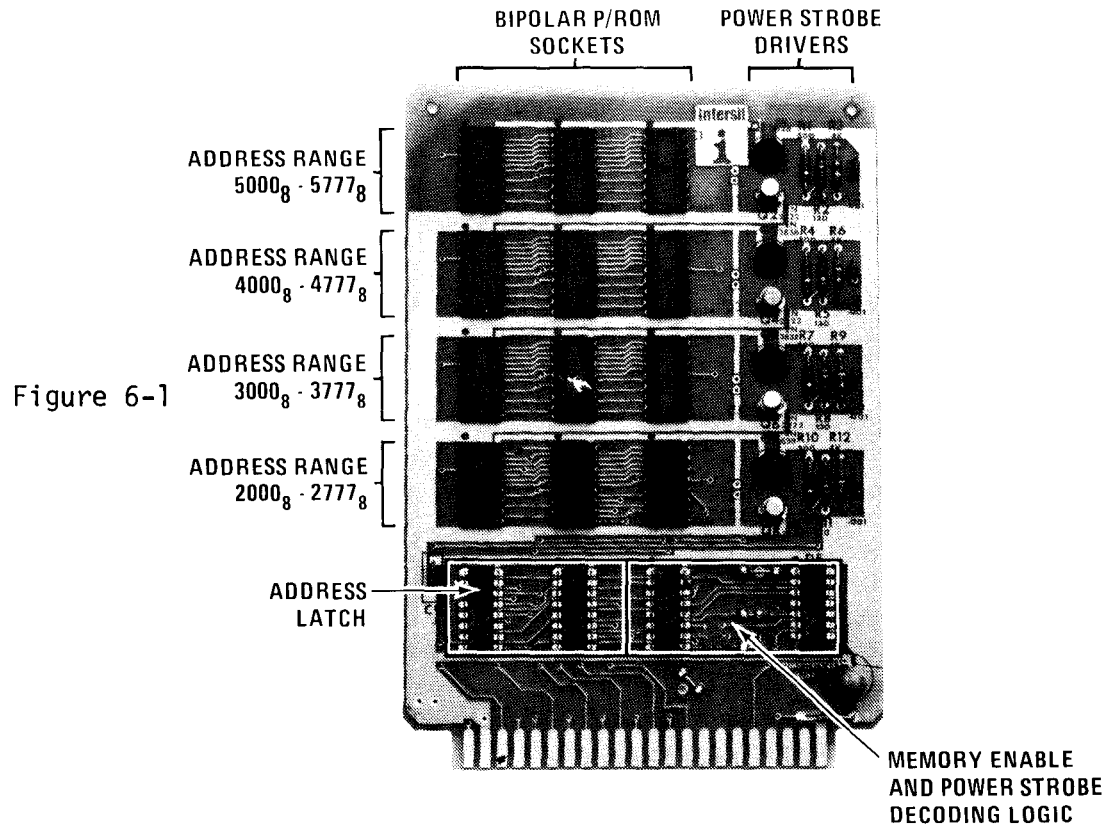
INTERSIL

JR. RAM MODULE SCHEMATIC

CHAPTER 6
JR. P/ROM MODULE

INTRODUCTION

The JR. P/ROM MODULE, 6952-P2KX12, pictured in Figure 6-1, enables user developed programs to be stored in user programmable read only memory.



The user has the option of utilizing the IM5623, 256 X 4, or IM5624, 512 X 4, three-state output Avalanche Induced Migration (AIM) programmable bipolar P/ROMs to obtain from 256 to 2048 words of program. Power dissipation is minimized by supplying power, via the POWER STROBE DRIVERS, only to those P/ROMs which are enabled. ADDRESS LATCH, MEMORY ENABLE AND POWER STROBE DECODING LOGIC are pictured in Figure 6-1.

The figure shows the address range for IM5624, 512 X 4 P/ROMs. For the user's convenience, the address range for the IM5623, 256 X 4, P/ROM and IM5624 are shown in TABLE 6-1. The user should change address range, as required, when mixing IM5623 and IM5624 on a given module.

TABLE 6-1

ADDRESS RANGE IN OCTAL IM5623/IM5624

<u>IM5623 (256 X 4)</u>	<u>IM5624 (512 X 4)</u>
2000-2377	2000-2777
3000-3377	3000-3777
4000-4377	4000-4777
5000-5377	5000-5777

DISCUSSION

This text should be used in conjunction with the enclosed schematic for a complete understanding of the 6952-P2KX12 JR. P/ROM MODULE.

The memory address is latched from the DX bus by the two 74LS174 hex latches when they are strobed by LXMAR.

The lower nine bits of the address go to the address inputs of all the twelve P/ROMs, which are arranged in a matrix of four rows of three.

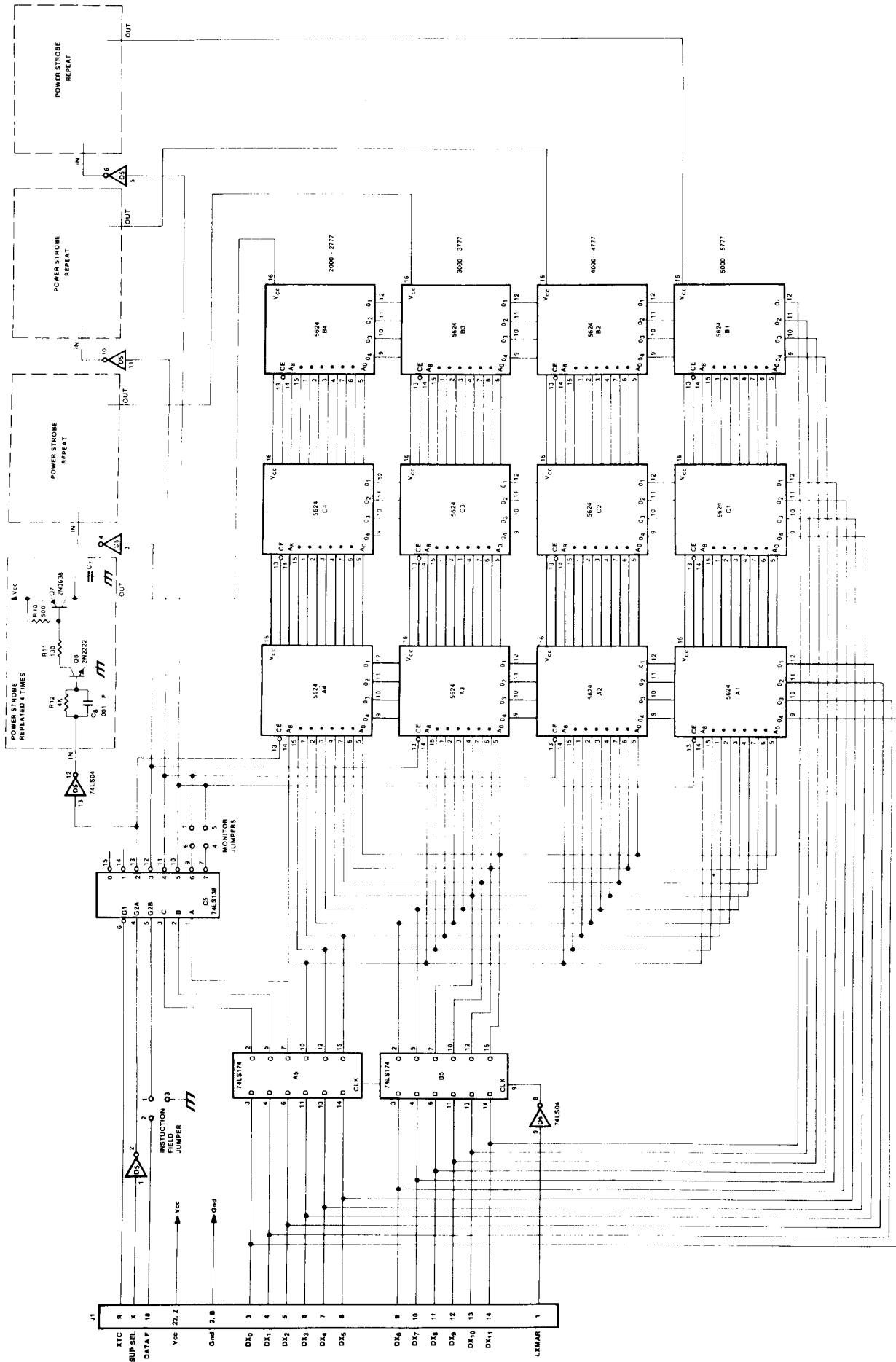
The higher order three bits of the address are decoded by the 74LS138, and it generates a chip enable to the appropriate row of P/ROMs. This chip enable is also used to turn on the two transistors in the appropriate power strobe circuit in order to connect V_{CC} (less a $V_{CE(SAT)}$) to the power pins of the enabled row of P/ROMs. There is no delay penalty in power strobing because the bipolar P/ROMs are much faster than required by the CMOS processor. The average power dissipation is reduced to approximately 5% of the non-strobed case. With the chip enable high, the P/ROM outputs are in a high impedance state permitting XTC to be used as one of the signals enabling the 74LS138 decoder. The P/ROM outputs, therefore, may be directly connected to the DX bus. The XTC line signals the read and write phases of the memory cycle. Thus, XTC when high, enables decoder pin G1 during the time that the address is latched into the 74LS174's, and remains enabled during the time the address is decoded, the P/ROMs are enabled, strobed and accessed. XTC goes low during the second half of the memory cycle, disabling the P/ROMs.

Decoder pin G2A is enabled only during the $\overline{SUP SEL}$ time, that is, when either MEMSEL or CPSEL is active. Therefore, the memory is really powered only for three clock cycles.

The uppermost 1K of memory is in the monitor ROM on the processor board, so the decoder does not use the pins for a decoded zero and one.

In the event that extended memory is used, the DATAF (DATA Field) pin is jumpered to the G2B enable pin of the 74LS138 decoder. This signal is normally low, enabling the decoder, and is activated to the high state during the executive phase of indirectly addressed AND, TAD, ISZ and DCA instructions (see IM6100 data sheet) so that data transfers are controlled by the Data Field, DF, and not the Instruction Field, IF, when addressing more than 4K words. Otherwise, the G2B pin may be left grounded by a jumper.

Table 6-1 shows the address space occupied by the P/ROMs. The user must supply at least three P/ROMs and can use them anywhere in the address space provided.

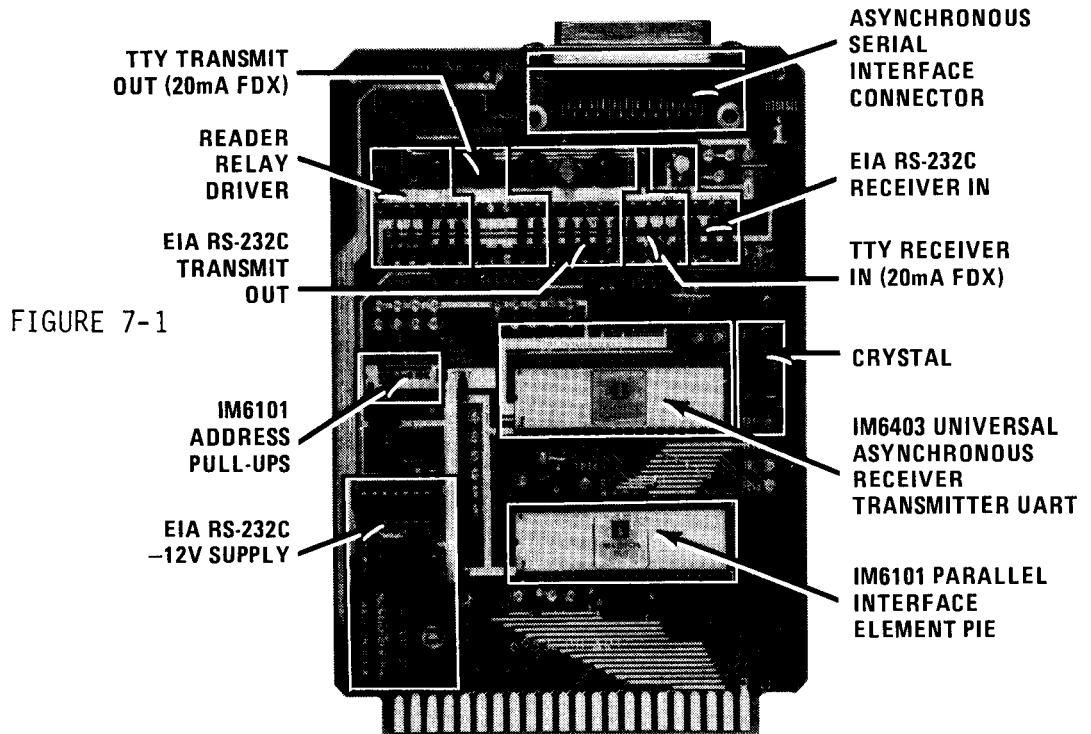


JR. P/ROM MODULE SCHEMATIC

CHAPTER 7
JR. SERIAL I/O MODULE

INTRODUCTION

The JR. SERIAL I/O MODULE, 6953-PIEART, pictured in Figure 7-1, allows the user to communicate with a 110 baud full duplex terminal with either an EIA RS-232C type differential voltage interface or a 20mA current loop interface.



This board uses two CMOS LSI chips, the IM6101 Programmable Interface Element (PIE) and the IM6403 Universal Asynchronous Receiver/Transmitter (UART). The MONITOR ROM provided with the 6950-INTERCEPT JR. MODULE contains a bootstrap loader for loading programs from the 6953-PIEART using BIN formatted media, such as paper tape punched out by the 6950-INTERCEPT JR. via the 6953-PIEART and an ASR-33 Teletype using the Memory Dump routines contained in the MONITOR ROM. This allows the user to create programs, dump them out on paper tape and use them at a later date by simply reading the tape back in.

DISCUSSION

The data sheets on the PIE and UART should be studied in order to fully understand the description of the operation of this module.

It will also be beneficial to study the listing of the PIE-UART routines in the MONITOR ROM.

The PIE address used is 00111, therefore, all IOT instructions to the PIE are of the form 616X or 617X in octal.

By using a UART, the amount of code required to do serial I/O is considerably reduced because bit timing is taken care of by the UART. Also, the programs become insensitive to the CPU clock frequency. Both the PIE (B3) and the UART (B1) are general purpose programmable devices and, therefore, need to be programmed or initialized to specific system requirements.

Some functions are programmed by hardwired pin connections and others by MONITOR ROM firmware routines.

The printed wiring is set up to program the PIE SEL 3-7 inputs to the address 00111. It also grounds CNTRL pin 2 of the 6403 UART selecting the internal 11 stage divider. This divider's output is the 16X clock used by the receiver register and transmitter register. The 6403 is designed to be directly clocked by a crystal. The crystal used is a TV colorburst crystal of 3,579,545 Hz. When this is divided by 2^{11} and 16, the baud rate of 109.2 Hz is within the tolerance limits of a 110 baud Teletype interface. The DIP package of 10K resistors (A3) pulls up the SEL 5, 6, 7 inputs and the PIE series priority input pin 3. The PIE control registers A and B and the vector register are initialized by the INPIE routine in firmware. Table 7-1 shows the constants loaded into these registers.

TABLE 7-1
CONTROL REGISTER A

0	1	2	3	4	5	6	7	8	9	10	11
FL4	FL3	FL2	FL1	WP2	.	WP1	.	IE4	IE3	IE2	IE1
1	1	1	0	1	0	0	0	0	0	0	0

- FL 2, 3, 4 bits set high cause the unused FLAG outputs 2, 3, 4 to be at high level
- FL 1 bit set low causes FLAG output 1 (Reader Run Relay Flag) to be at low level
- WP 2 set high means positive WRITE POLARITY or positive pulses at WRITE output 2 (used to load the UART CONTROL REGISTER)

WP 1 set low causes negative pulses at WRITE output 1 (used to load the UART TRANSMITTER BUFFER REGISTER from the data inputs).

IE 1, 2, 3, 4 set at 0 disables all PIE interrupts.

TABLE 7-2
CONTROL REGISTER B

0	1	2	3	4	5	6	7
SL4	SL3	SL2	SL1	SP4	SP3	SP2	SP1
0	0	1	1	0	1	1	1

NOTE:

1. Sense input S4 is not used, therefore, SL4 and SP4 bits are irrelevant.
2. SL 3 = 0 and SP 3 = 1 program the SENSE3 flip flop to be set by a positive going edge. SENSE3 is connected to the serial data input of the UART and is used for start bit detection.
3. SL2 = 1 and SP 2 = 1 program the SENSE2 flip flop to be set by a high level. SENSE2 is connected to the TRANSMITTER BUFFER REGISTER EMPTY (TBRE) output of the UART which indicates that the UART transmitter is ready for new data. The TBRE signal is a high level.
4. SL 1 = 1 and SP 1 = 1 program the SENSE1 flip flop to be set by a high level. SENSE1 is connected to the DATA READY (DR) output of the UART, which is a high level indicating that a character has been received and transferred to the receiver buffer register.

TABLE 7-3
VECTOR REGISTER

0	1	2	3	4	5	6	7	8	9	10	11
INTERRUPT VECTOR										VPR1	
0	0	0	0	0	0	0	0	0	0	0	0

NOTE: The PIE interrupts are disabled in this application, and the sense flip flops are tested by the firmware with SKIP instructions.

The PIE's READ2 output is unused and the READ1 output is connected to the UART RECEIVER REGISTER DISABLE (RRD) and DATA RECEIVED RESET (DRR, an active low input) so that when a received character is ready, R1 which is normally high (keeping the RECEIVER REGISTER disabled) pulses low during IOTA-DEVSEL, transferring the receiver data to the IM6100 via the DX bus while simultaneously clearing the DR flag in readiness for the next character.

The UART is also initialized both via hardwired connections and under program control.

STATUS FLAGS DISABLE (SFD pin 16) is grounded to enable all UART status flags. The UART CONTROL REGISTER bits are loaded from the DX bus as shown in Table 7-4.

TABLE 7-4

DX Lines	0	1	2	3	4
Designations	PI	SBS	EPE	CLS1	CLS2
Constant	1	1	1	1	1

PI = 1 PARITY INHIBIT - Parity generation and checking is inhibited and PARITY ERROR (PE) output is forced low.

SBS = 1 STOP BIT SELECT - In conjunction with CLS1 and CLS2, this selects two (2) stop bits.

EPE = 1 EVEN PARITY ENABLE - Irrelevant as parity is inhibited.

CLS1 = 1)
CLS2 = 1) CHARACTER LENGTH SELECTED - These bits select on eight-bit character.

All unused pins are brought out to test points, to facilitate experiments by the user.

The UART TBR parallel data input bus and RBR parallel data output bus are connected to DX4-11.

The serial input and output pins of the UART go to both EIA-RS-232C and 20 mA current loop interface drivers and receivers.

Table 7-5 shows the connector and jumper options for the two interfaces.

Serial output bits from the UART cause the push-pull EIA driver to switch between V_{CC} and -12 volt and transistor Q2 to supply 25 mA nominally ($5 \text{ volt} \div (R5 + R4)$) to the current loop interface.

Briefly, the PIEART interface works as follows once the interface is initialized. When transmitting to a terminal, the IM6100 executes a waiting loop using a SKIP on SENSE2 instruction followed by a jump back. SENSE2 as shown in Table 7-3 is set when the TRANSMITTER BUFFER is empty. When the character has been transmitted, the waiting loop is exited and a WRITE1 instruction is executed writing a new character into the UART transmit buffer. The PIE strobes the DX bus at the proper time when this instruction is performed.

When receiving from a terminal, the IM6100 resets the SENSE3 flip flop by executing a SKIP on SENSE3 instruction. This flip flop senses the start bit of a character. The READER RUN flag is set by executing a SET FLAG 1 instruction to the PIE. Now the interface is ready for a character from either a tape reader or a keyboard and a wait loop is entered. This loop is exited when a start bit is detected and the READER RUN flag is cleared just in case the data source was a reader. This stops the reader from advancing until the CPU is ready for another character. Another wait loop is entered and this time it is exited when the DATA RECEIVED flag goes true, setting the SENSE1 flip flop. The accumulator may then be cleared and a READ1 command executed. This causes the PIE to enable the UART receiver buffer onto the DX bus, simultaneously clearing the DR flag.

When reading BIN tape, the above transmit and receive program sequences are called as subroutines, while the main program performs functions such as testing characters for a rubout, accumulating checksums, testing for leader-trailer, etc. (Refer to MONITOR description).

Whenever SKIP on SENSE flip flop instructions are executed, the PIE will test the state of the desired flip flop and, if it has been set, it will assert the SKP/INT output causing the IM6100 to skip the next instruction. The sense flip flop is then cleared. For more details, refer to the PIE data sheet.

TABLE 7-5
20 mA LOOP/EIA RS232-C CONNECTOR PINOUTS

OPTION	STANDARD CONNECTION	MODIFIED CONNECTION
Voltage Change Option	+5 VDC on V _{CC} Connect points #1 and #2	+10 VDC on V _{CC} Cut between points #1 and #2 and connect points #1 and #3
Driver/Receiver Change Option	20 mA loop Connect points #4 and #5	EIA RS232-C Cut between points #4 and #5 and connect points #5 and #6
EIA Earth Ground Option	No EIA Earth ground	To connect Earth ground, tie points #7 and #8 together

CONNECTOR PINOUTS

20 mA Loop		EIA RS232-C	
<u>Pin</u>	<u>Signal</u>	<u>Pin</u>	<u>Signal</u>
1	XMIT+	1	Earth Ground
2	KEY	2	XMIT
3	XMIT-	3	RCVE
4	RCVE+	7	Signal Ground
5	RCVE-	18	-12 VDC
6	RDR+	All others are N.C.	
7	RDR-	Pins 5 (Clear to Send)	
8	-12 VDC	6 (Data Set Ready)	
9	N.C.	8 (Received Line Signal Detector)	
10	N.C.	may have to be tied to VCC with some terminals	

In order to use the module, it must first be connected to a serial ASCII 110 baud tape reader, typically an ASR-33 Teletype equipped with the reader. The connection is done by a cable connecting the 20 mA loop connector pins to the Teletype terminal strip. The Teletype is turned to the LINE position.

Note that the Teletype must be equipped for 20 mA full duplex operation and should have a reader run relay installed (such as 6909-RELAY).

To read BIN format tape, the tape is placed in the reader, the key is put in the START position and the sequence CNTRL 1 is pressed on the INTERCEPT JR.

As explained on page 2-5, this function will activate the loader. At the end of the load sequence, the machine is halted showing the AC (SAVAC location 0140) whose contents represent the checksum and should be zero for a valid load.

To dump memory onto tape, the starting and ending address of the block should be entered into locations 0176 and 0177 and the program run starting at location 7510. Naturally, the tape punch should be turned on.

Chapter 8 page 14 describes these routines in more detail.

Table 7-6 lists the PIE-UART instructions as used by the MONITOR. These instructions are also listed in the program listing.

TABLE 7-6
PIE-UART INSTRUCTIONS

6160	READ1	(Reset UART Data Received Flag and read received character)
6170	READ2	(Generate read strobe 2) - Not used
6161	WRITE1	(Load UART Transmit Buffer)
6171	WRITE2	(Load UART Control Register)
6162	SKIP1	(Test state of sense FF1; skip if set by UART Data Received Flag)
6163	SKIP2	(Test state of sense FF2; skip if set by UART Transmit Buffer Empty Flag)
6172	SKIP3	(Test state of sense FF3; skip if set by START bit)
6173	SKIP4	(Test state of sense FF4; skip if set) - Not used

6164	RCRA	(Read control register A)
6165	WCRA	(Write control register A)
6175	WCRB	(Write control register B)
6174	WVR	(Write vector register)
6166	SFLAG1	(Set FLAG 1) - Reader Relay Flag - ON
6176	SFLAG3	(Set FLAG 3)
6167	CFLAG1	(Clear FLAG 1) - Reader Run Relay Flag - OFF
6177	CFLAG3	(Clear FLAG 3)

In addition to these, the IM6100 internal IOT instruction 6007g or CAF (Clear All Flags) clears the sense flip-flop thus clearing all interrupt requests.

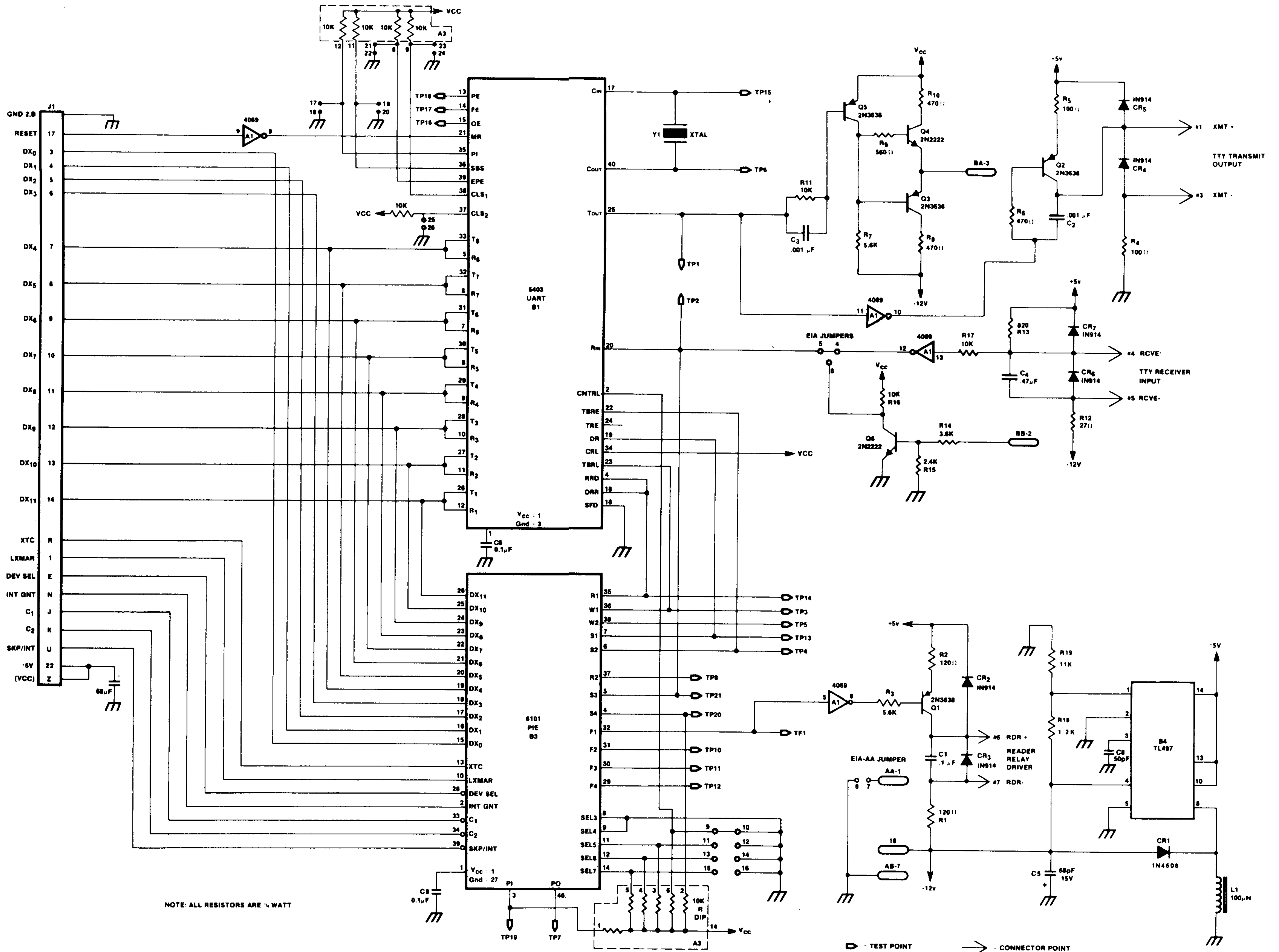
The serial I/O module is typically used with the INTERCEPT JR. BINARY LOADER and MEMORY DUMP routines in order to read BIN format tape and dump a block of memory onto BIN formatted tape.

The PIE-UART interface is initialized only when the BIN and DUMP programs are used. The user has access to these routines via the software subroutine call stacking mechanism in case the serial port is to be used for other purposes, such as printing characters on the Teletype.

The user may also write his own code in RAM for interface utilization and handling Teletype I/O.

Example 14 in Chapter 3 shows how the MONITOR subroutine may be called to implement Teletype keyboard and printer operation.

JR. PIEART SERIAL I/O MODULE SCHEMATIC



NOTE: ALL RESISTORS ARE 1/4 WATT

CHAPTER 8
INTERCEPT JR. TUTORIAL SYSTEM MONITOR PROGRAM

The MONITOR is structured as an interrupt driven main program refreshing the display and looking for a CNTRL key depression; upon detecting it, it branches to a routine SHELL that picks up the next key depression, branches to appropriate routines and performs the operation.

The MONITOR uses main memory to store control panel routines in order to keep the system inexpensive. The IM6100 architecture, however, will allow control panel programs to exist in separate memory totally transparent to the user.

Figure 8-1 shows the memory allocation map for INTERCEPT JR.

The MONITOR uses several locations in page 0. These are listed in the program.

Some of these locations, SAVAC, SAVMQ, SAVFL in location 0140g, 0141g, 0142g, are used by the MONITOR to store IM6100 registers and flags and enable the user to conveniently examine and alter these registers.

Locations 0000 and locations 0143g to 0177g inclusive may not be altered by entering data through the keypad (MEM, DECPC or MICRO modes) or by using the BIN loader.

The user is urged to follow the descriptions of the MONITOR routines by referring to the program listing. The symbol table at the end of the listing may be used to find subroutine entry points and absolute addresses of symbolic operands.

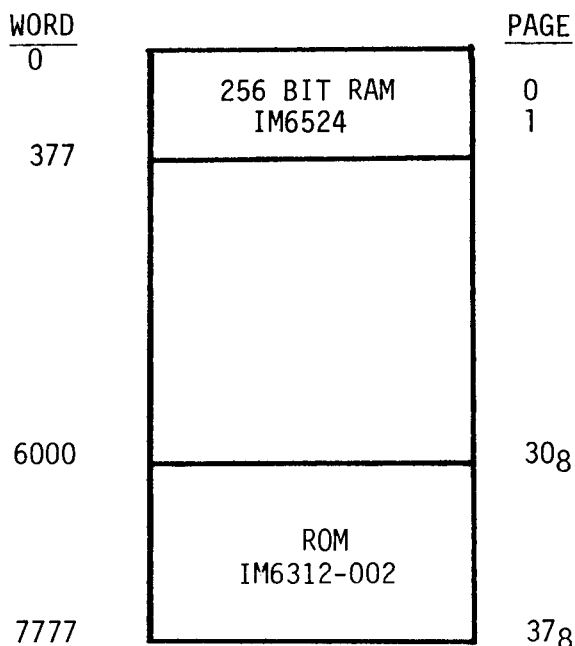
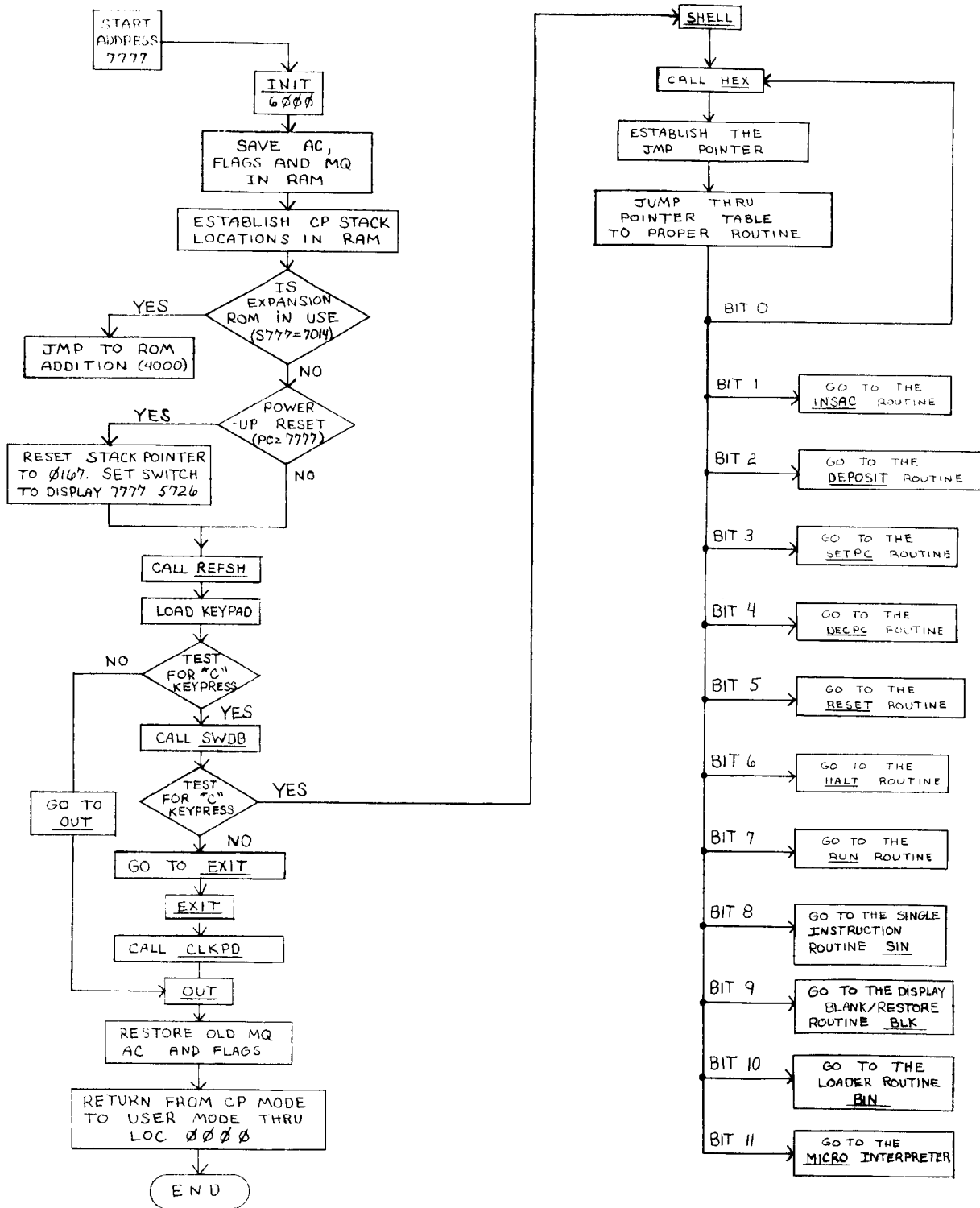
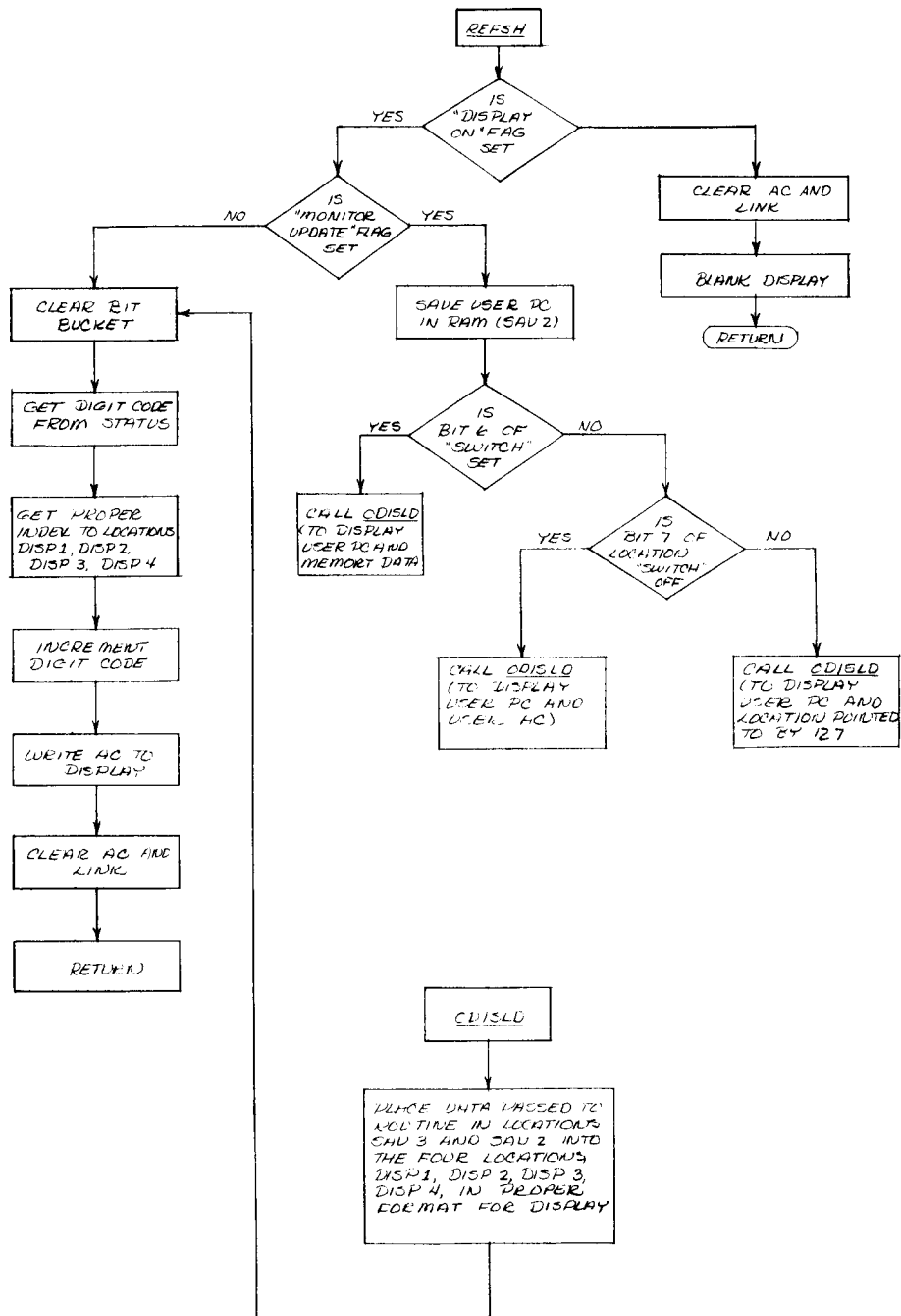
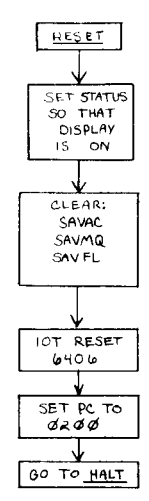
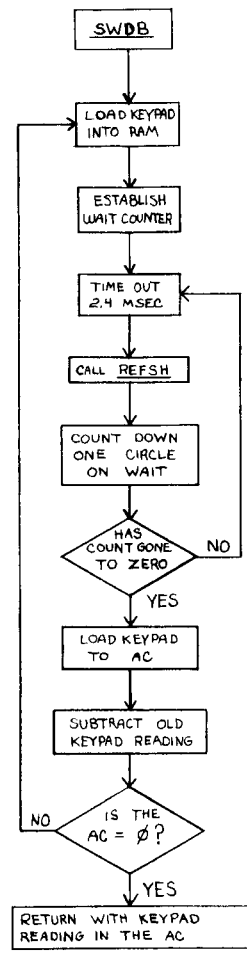
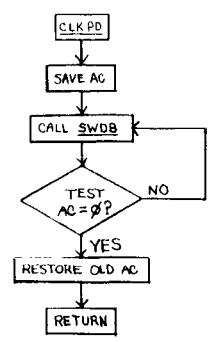
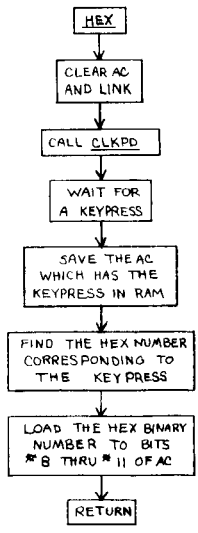


FIGURE 8-1

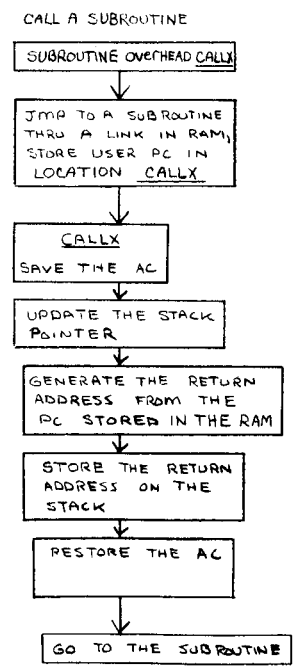
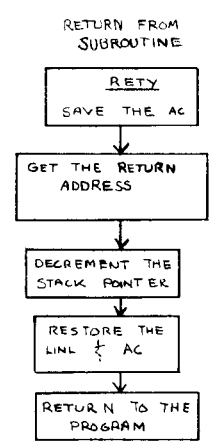


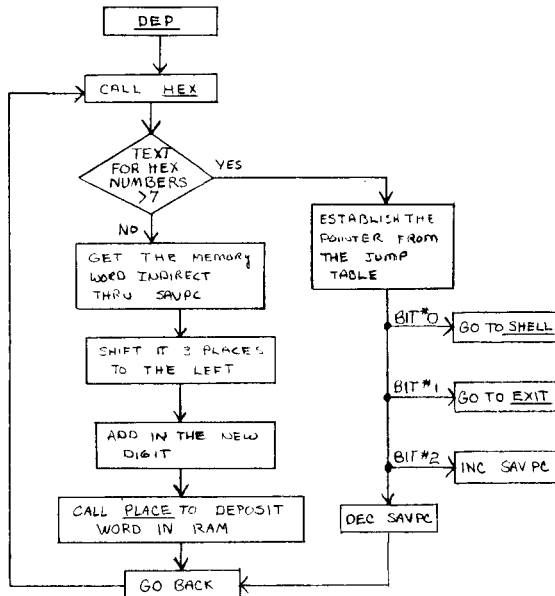
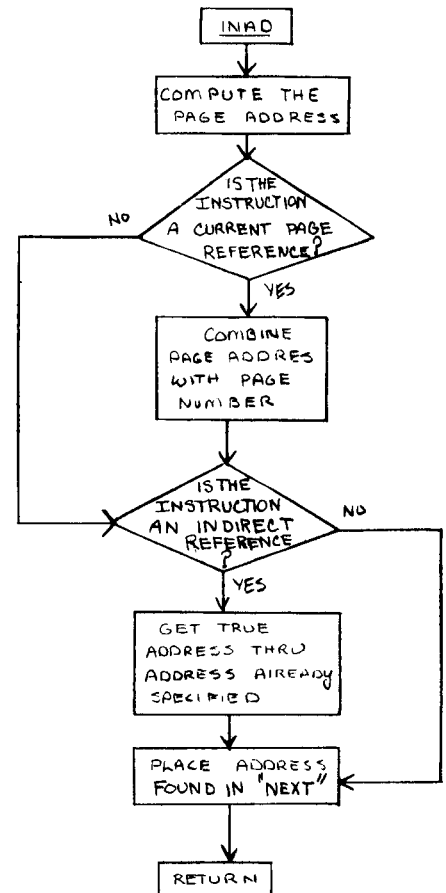
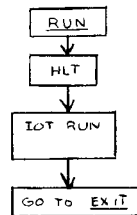
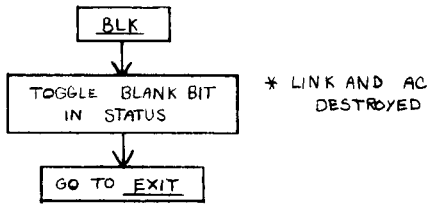
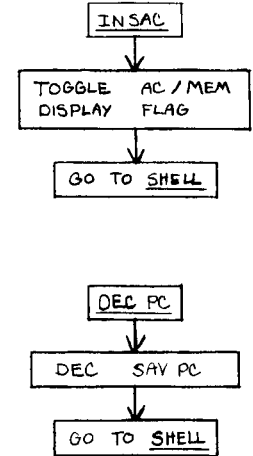
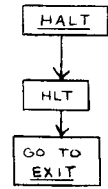
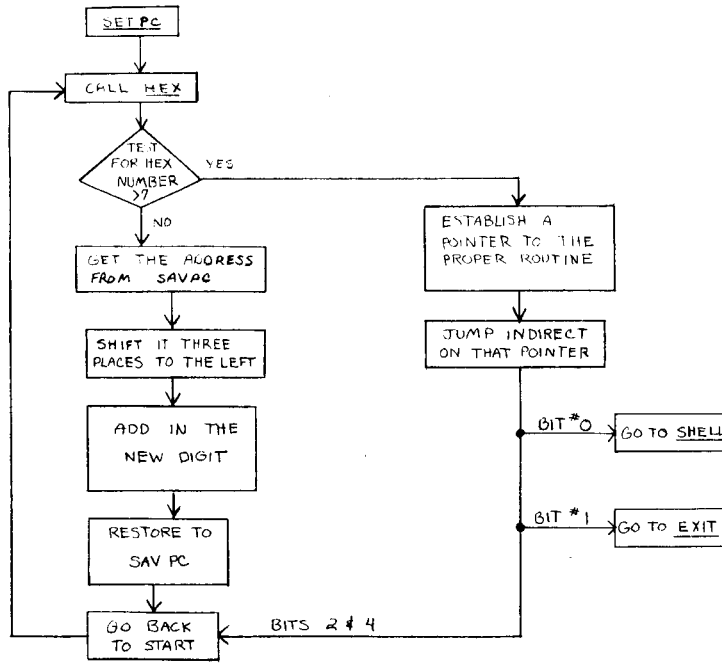


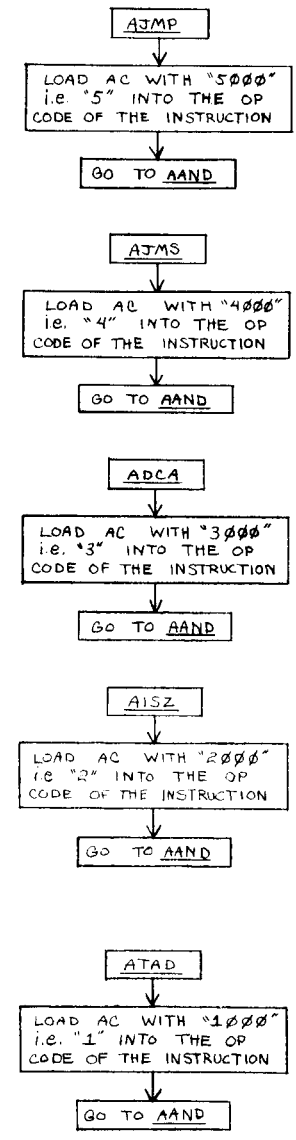
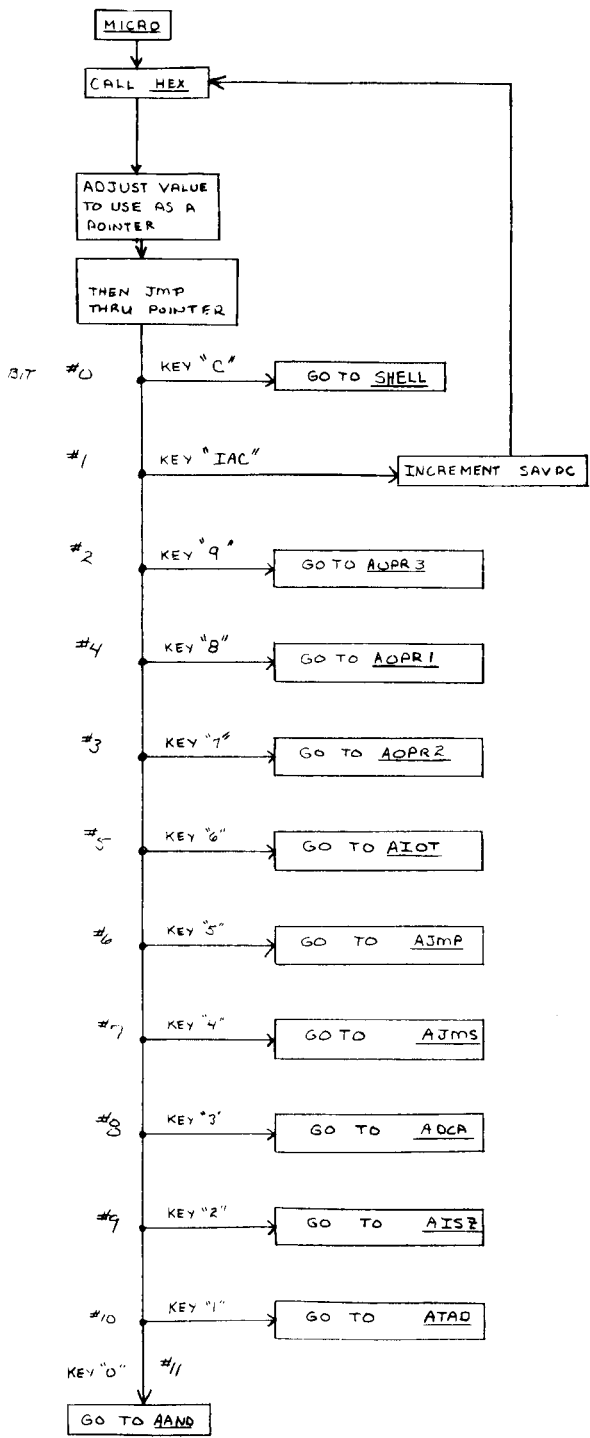


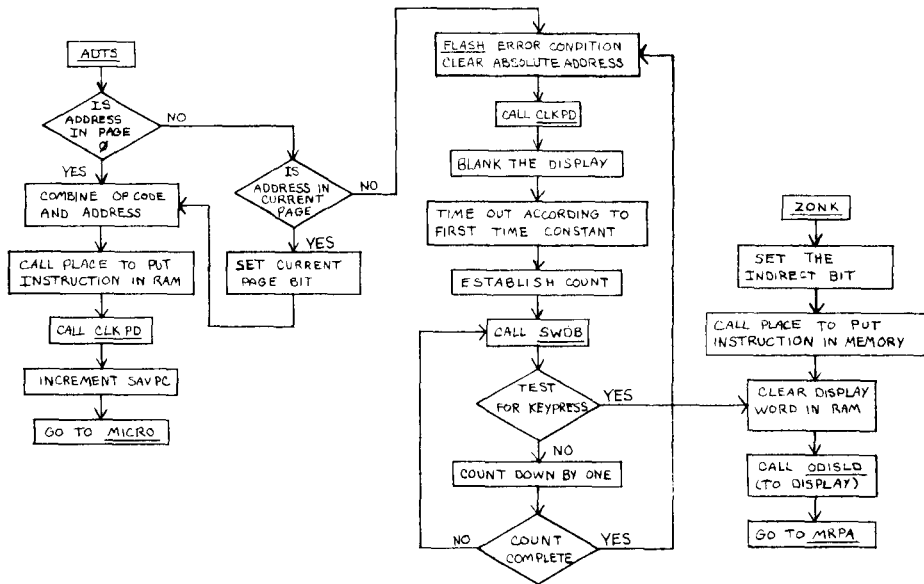
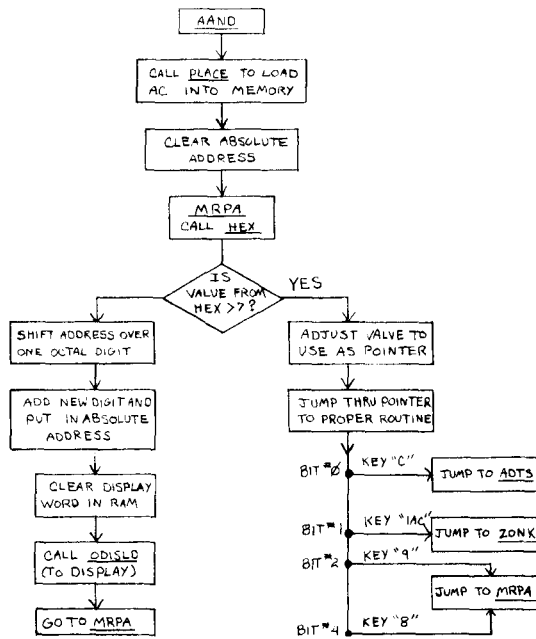
VALUE TABLE FOR HEX

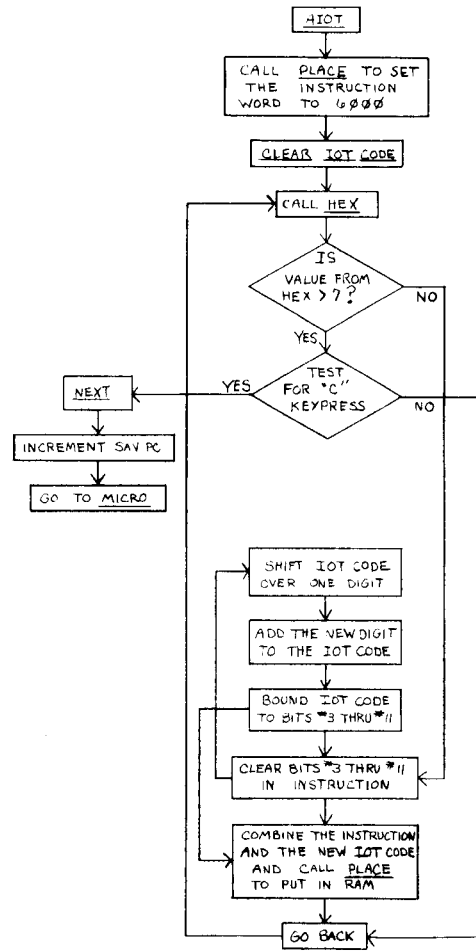
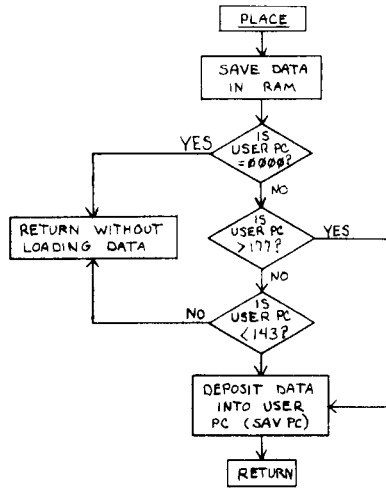
BUTTON	SW REG BIT #	HEX VALUE
RED	0	0
YELLOW	1	A
"MEM"	2	9
"DEC PC"	3	8
7	4	7
6	5	6
5	6	5
4	7	4
3	8	3
2	9	2
1	10	1
0	11	0

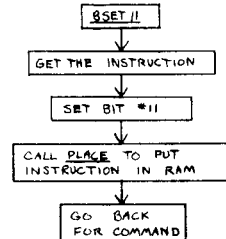
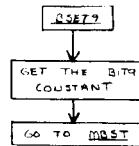
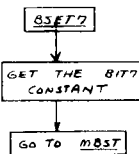
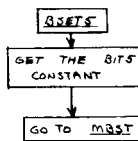
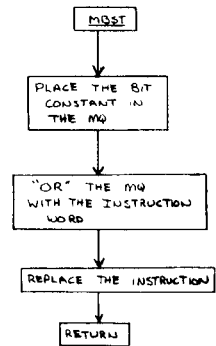
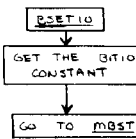
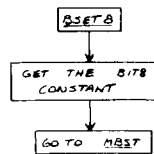
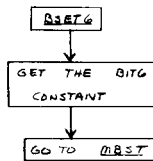
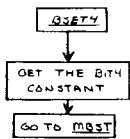
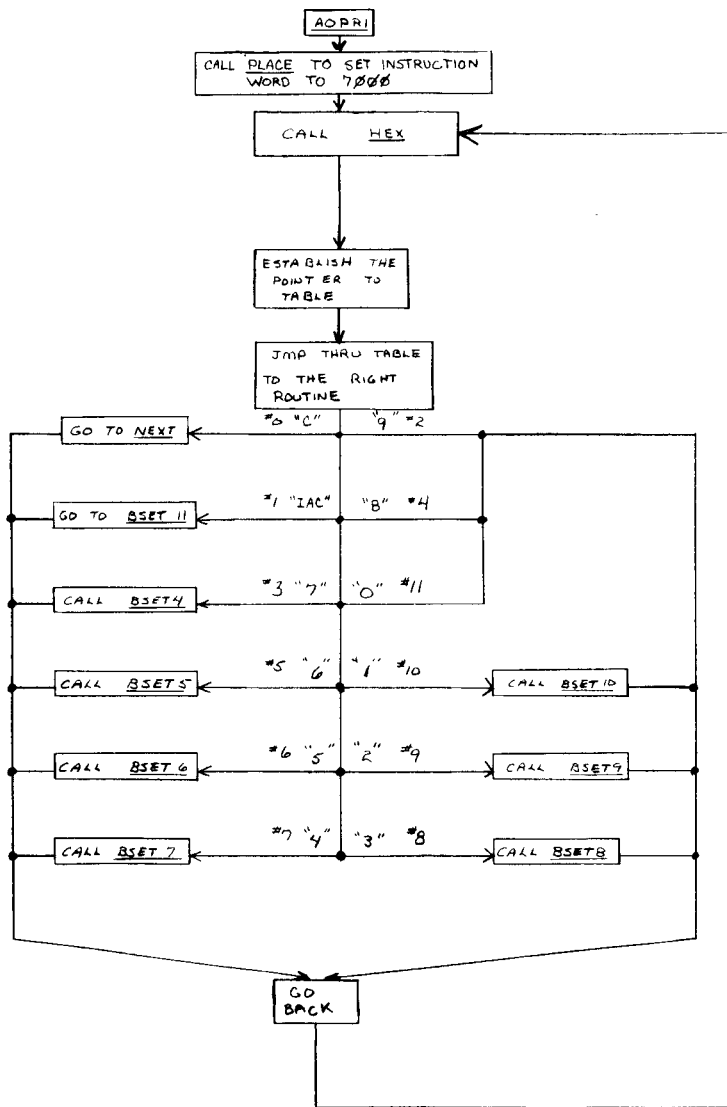


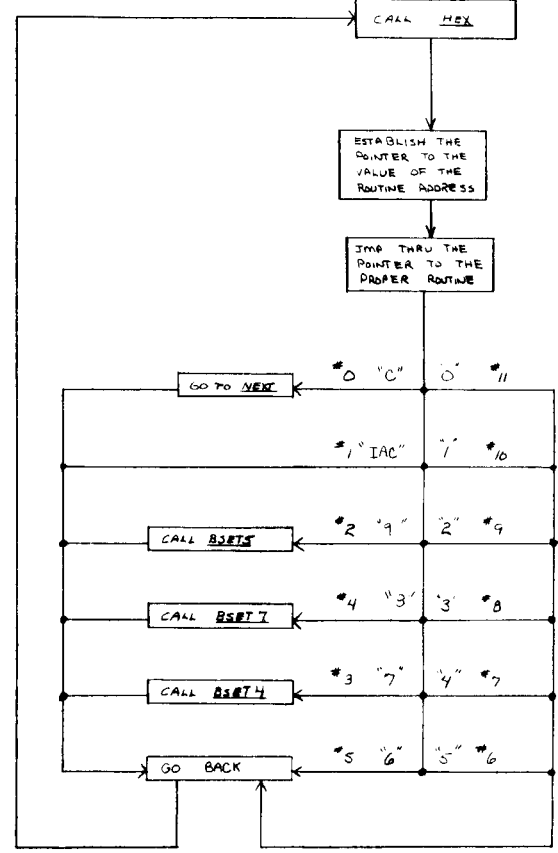
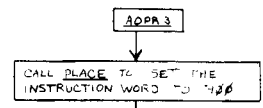
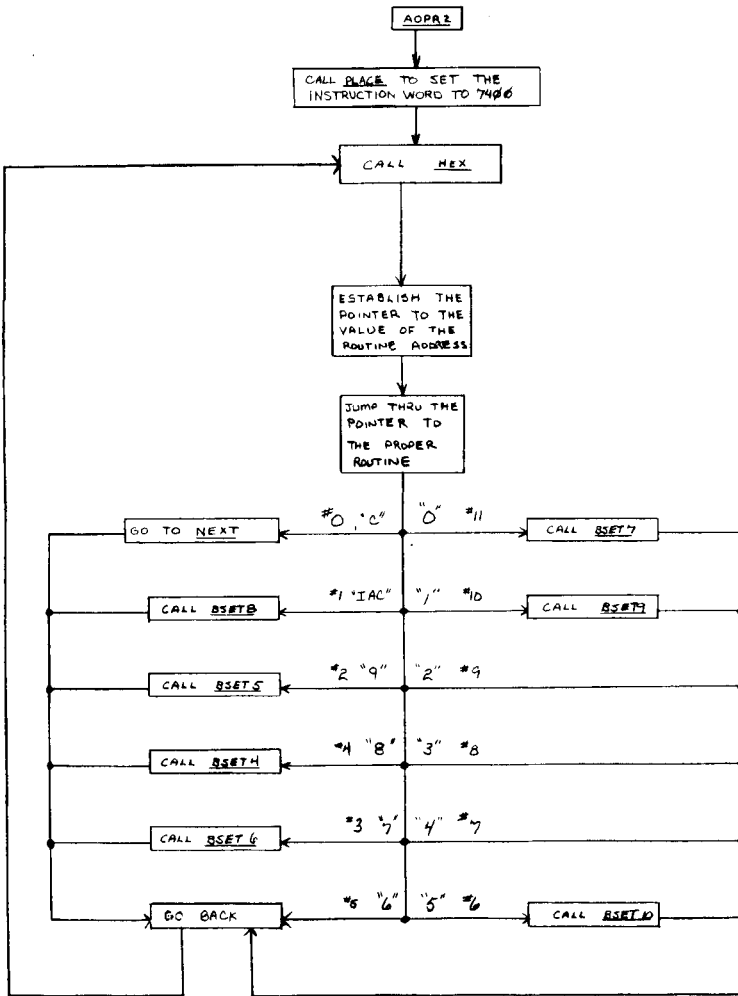


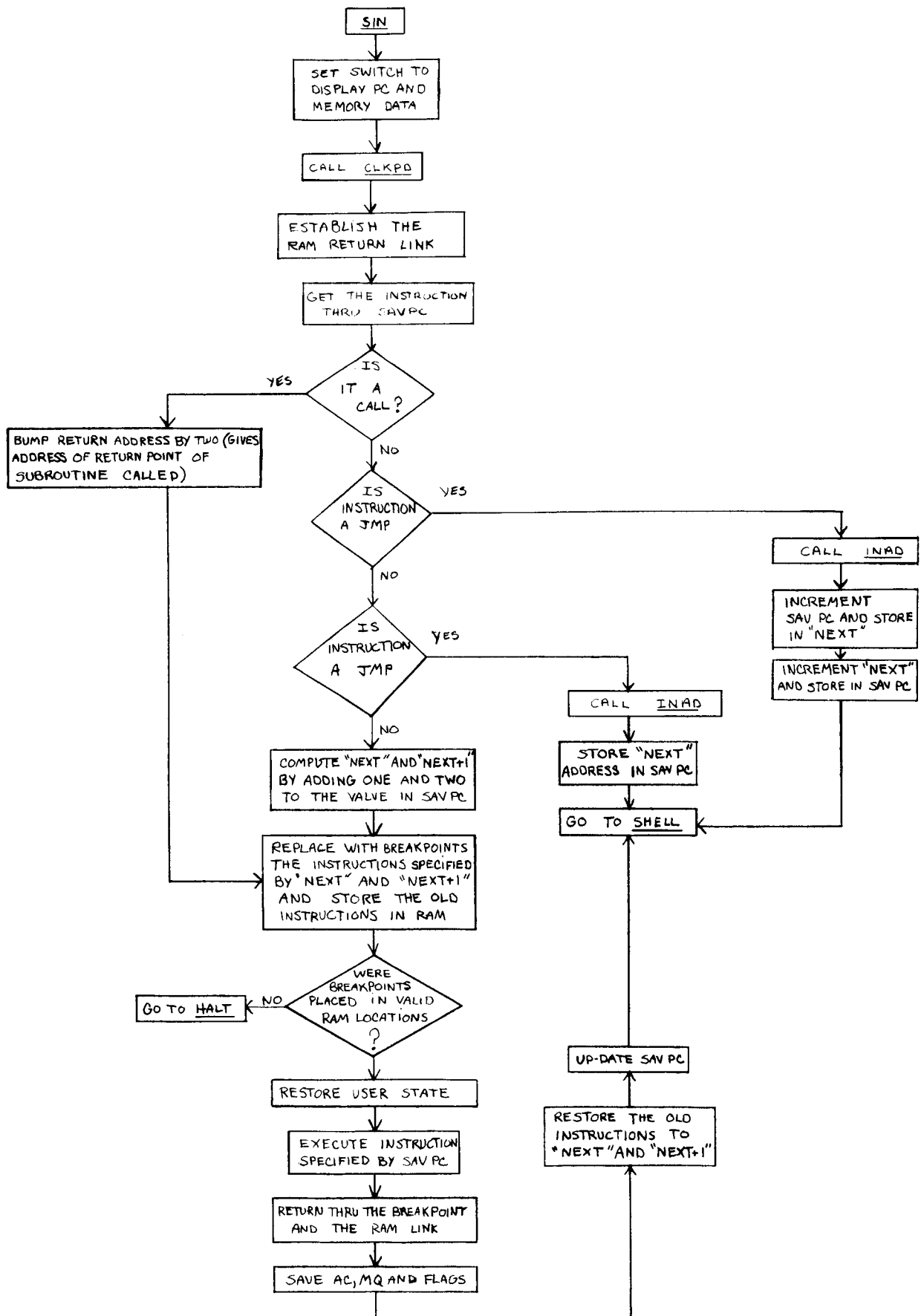


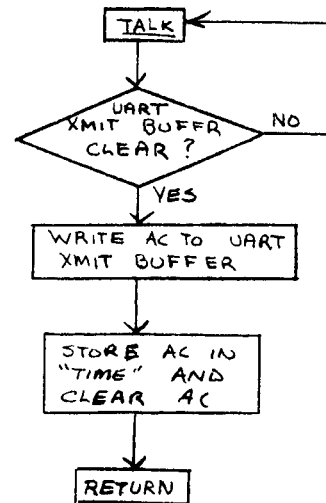
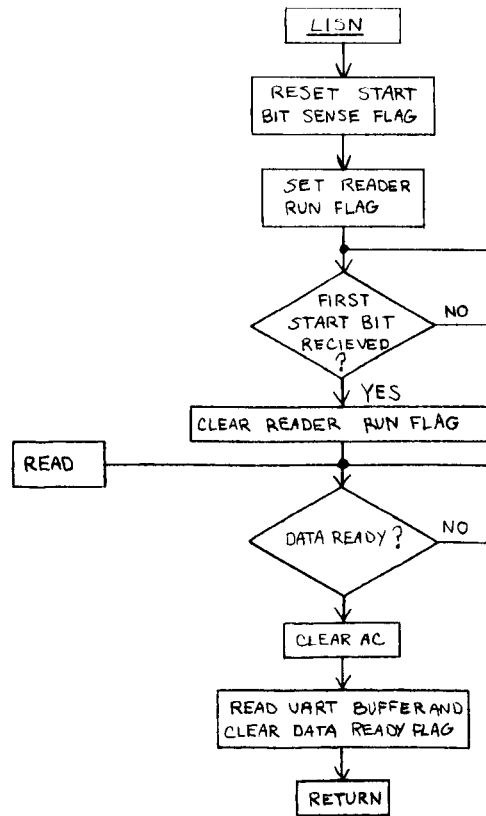
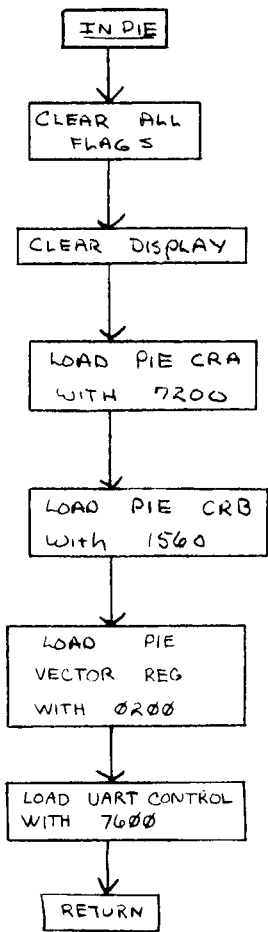


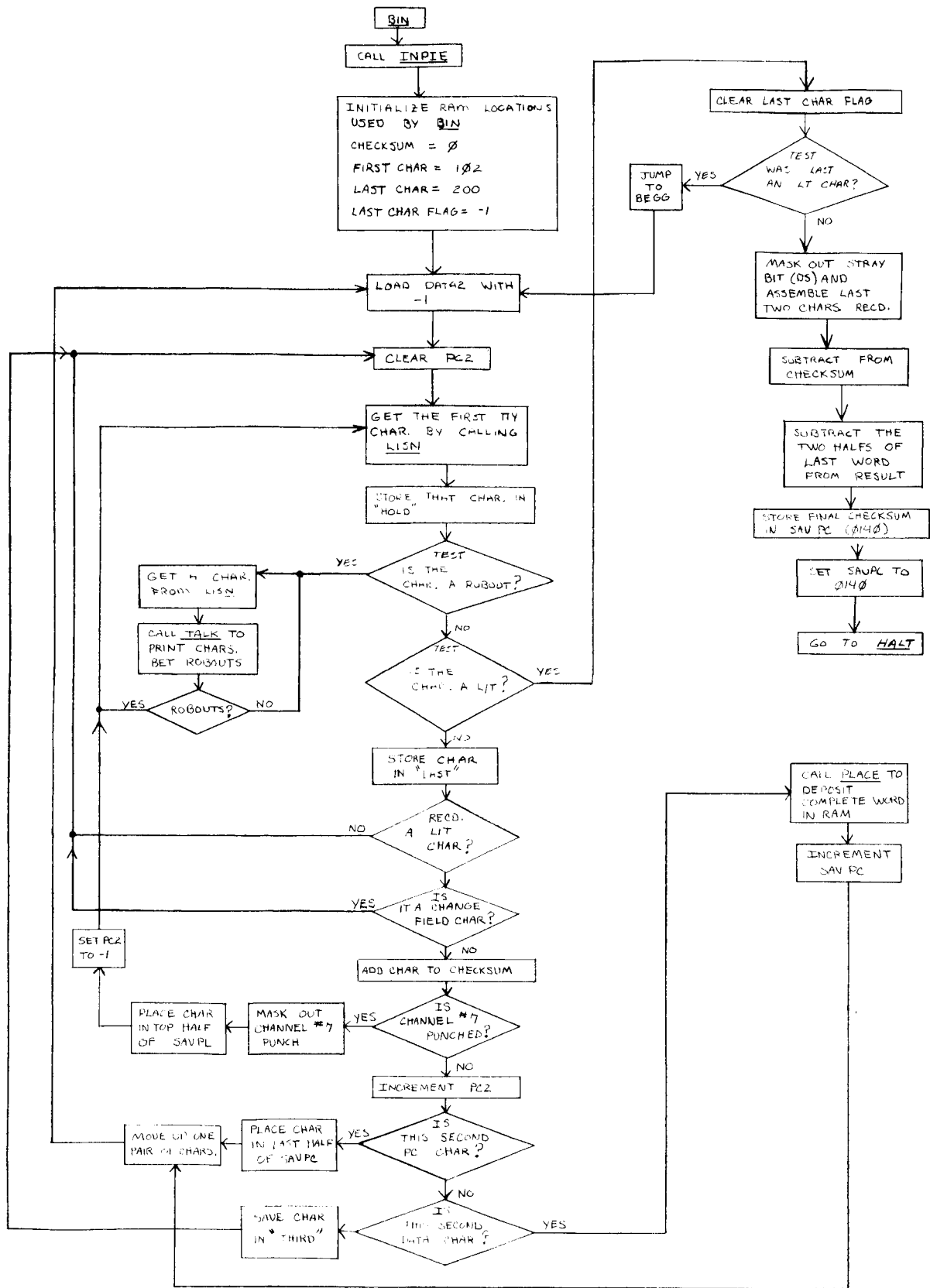


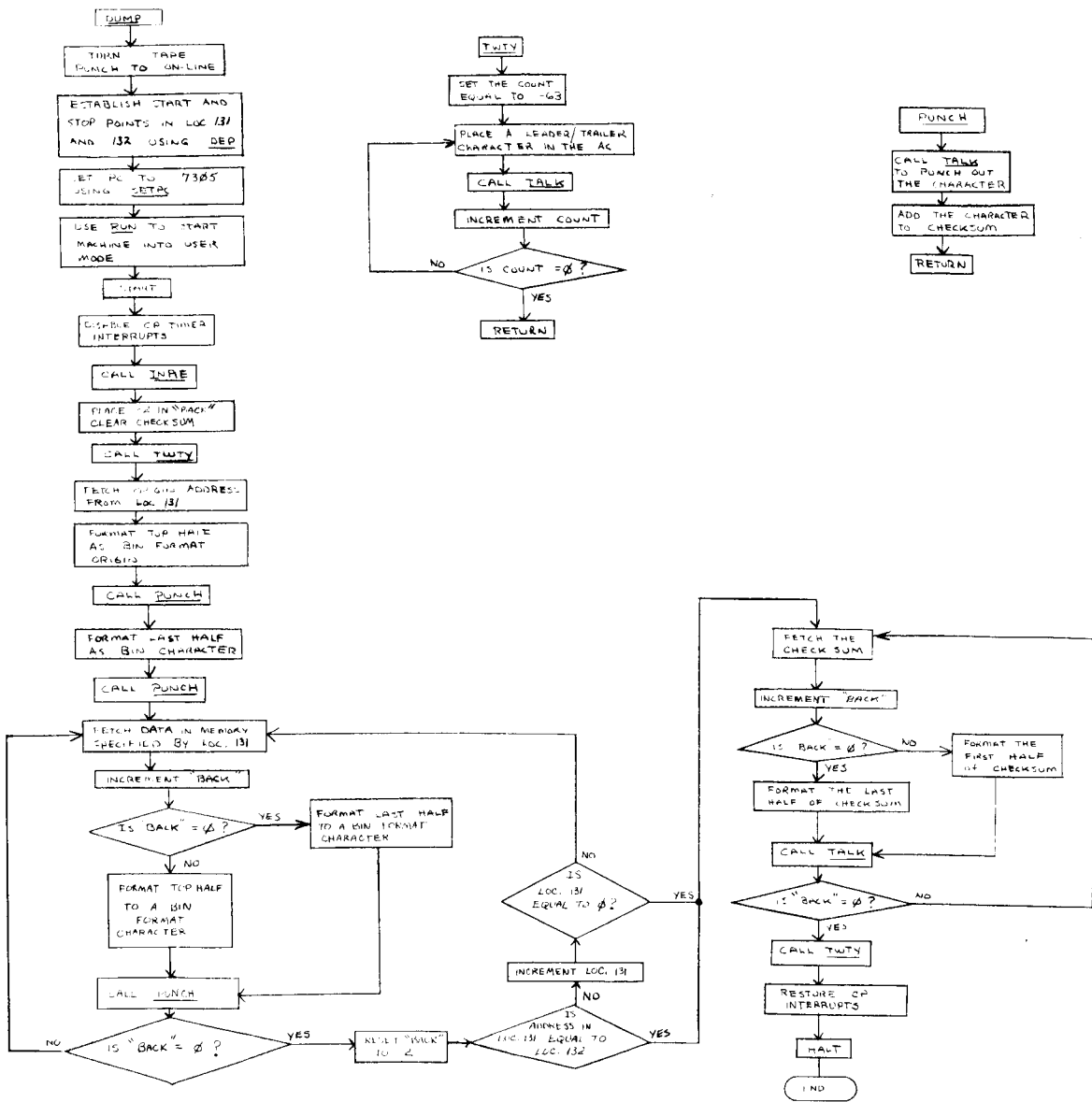












MONITOR STACK

Summary

Locations 167g to 177g are used as a software stack for subroutine return addresses. Additional area can be allocated to the stack by reserving any amount of space from 200g on down.

On every CP interrupt after saving AC, MQ and FLAGS, the MONITOR reestablishes the stack locations in RAM but will reset the stack pointer and display modes ONLY if the user's program counter is pointing to location 7777. A stack call is implemented by the instruction 4161 followed by a 12-bit absolute pointer address and a return is implemented by the instruction 5564.

Discussion

The JMS (jump to subroutine) instruction of the IM6100 operates by storing the return address in the location referenced by the instruction and stepping to the following location. This location must contain an executable instruction. ROM resident subroutines must have their entry points in RAM, as ROM cannot be written into. The MONITOR uses a pushdown stack to minimize the overhead involved in storing subroutine return addresses.

A subroutine is "called" by invoking a supervisory routine, CALL, followed by the subroutine entry address. CALL increments the PC then leaves it on a stack, starting at 0167, updating the stack pointer in 0165. A return from the subroutine is performed by executing another routine, RETURN, which links back to the main program by "popping" the return address off the stack, decrementing the pointer. The return address skips over the entry address which followed the CALL statement. By reserving enough space in RAM, subroutines may be "nested" to any practical depth desired. Programs starting at location 0200 limit the stack depth to nine locations, of which several may be used at any particular time by the MONITOR routines. The program makes no provision for interrupt service routines using the stack since these higher priority routines may overwrite locations used for temporary variables by subroutine calls or returns.

Referring to Page 8-2, the INTERCEPT JR. MAIN FLOW CHART, the MONITOR is entered on power-up or on every CPREQ through location 7777 of control panel memory and the return address is saved in location 0000. The MONITOR updates the register save locations and goes on to the initialization routines. The CP subroutine stack is established. (Refer to Appendix L for a description of software stack operation with the IM6100.) Returns from subroutine calls should normally leave AC, MQ and L unchanged.

Next, the presence of the expansion ROM is checked. If location 57778 has 07648 in it, the program branches to location 40008, which should be the entry point for the additional ROM.

If the expansion ROM is not present, the MONITOR checks whether it is going through a power-up RESET (PC = 77778). The stack base is initialized only if there is a power-up RESET or user PC is 77778.







The Display Refresh subroutine, REFRESH, is executed 100-200 times a second in order to keep the display flicker-free.

Next, the keypad is tested for depression of the CNTRL key. If this is not detected, the monitor goes to the out point, restores registers and flags and returns via the pointer in location 0000.

If a CNTRL key depression is detected, the switch debouce routine, SWDB, is called, and the test for CNTRL is made again. In case the test fails, the routine waits for the keypad to become inactive, by calling CLKPD, and exits as before. If the CNTRL key is definitely detected, the MONITOR enters the undefined control state SHELL and subsequent key depressions will have to be detected and analyzed. The MONITOR calls HEX, which generates starting addresses for the subroutines that are used to service each of the different key depressions that define a control state. Figure 8-4 shows the connections between the keys and the DX bus, and the control state selected by the key.

The MONITOR is directed to the proper service routine, and may or may not need further data (more key depressions, external conditions, status word bit settings, etc.) to properly execute the routine.

We shall now study some frequently called subroutines in the MONITOR ROM, REFRESH, SWDB, CLKPD, HEX and EXIT.

DX LINE	0	1	2	3	4	5
KEYBOARD						
CONTROL STATE	CNTRL	SHIFT	MEMory data deposit	SETPC	DECPC	RESET
VALUE RETURNED BY HEX	0013 ₈ or B ₁₆	0012 ₈ or A ₁₆	0011 ₈ or 9 ₁₆	0007 ₈ or 7 ₁₆	0010 ₈ or 8 ₁₆	0006 ₈ or 6 ₁₆







DX LINE	6	7	8	9	10	11
KEYBOARD						
CONTROL STATE	HALT	RUN	SINgle instruction execute	DISPlay blank/restore	binary loader	MICRO interpreter
VALUE RETURNED BY HEX	0005 ₈ or 5 ₁₆	0004 ₈ or 4 ₁₆	0003 ₈ or 3 ₁₆	0002 ₈ or 2 ₁₆	0001 ₈ or 1 ₁₆	0000 ₈ or 0 ₁₆

FIGURE 8-4

The REFSH routine checks the display flag (MSB) of the STATUS word in location 0143g. If the flag is cleared, the display is blanked. If the flag is set, the routine examines location 0133g, the SWITCH word. If the MONITOR UPDATE flag in the MSB of this word is clear, the routine jumps to UDIS. UDIS uses the display code in bits 10 and 11 of the status word as an index to one of the locations DISP1, DISP2, DISP3, DISP4 by adding these two bits to the constant TADJ - 0134g and using the sum as a pointer. Thus UDIS refreshes the "USER DISPLAY".

If the MONITOR UPDATE FLAG is set, the User PC is stored in SAV2 (0155g) and bit 6 of location SWITCH is tested. If this bit is set, the data at the User's PC is obtained, stored in SAV3 (0156g) and the ODISLD routine is called. This routine formats the contents of SAV2 and SAV3 into four words that are placed into locations DISP1, DISP2, DISP3, DISP4. These locations are used to update the display.

If bit 6 in SWITCH is cleared, the routine checks bit 7. If bit 7 is set, the User AC stored at SAVAC location 0140g is transferred to SAV3 so that the User PC and User AC are displayed in real time. The SHELL will recognize the CNTRL IAC sequence as a request to complement bit 6 in SWITCH.

If bit 7 is clear, location 0127g is used as a pointer to the word that will be placed in SAV3. On power-up initialization, SWITCH is loaded with 7777g so bit 7 is set at that time.

Figures 8-3A, 8-3B, 8-3C show the display options available to the user as determined by STATUS and SWITCH bits.

0	1	2	3	4	5	6	7	8	9	10	11
ST ₀	X	X	X	X	X	X	X	X	ST ₉	ST ₁₀	ST ₁₁

X = Don't Care

ST₀ = 0 - Displays blanked
 1 - Display refresh enabled

ST₉ = "bit bucket" catches carry out when ST₁₀, ST₁₁ are incremented. Program clears this bit before every update.

ST₁₀ ST₁₁ = DISPLAY CODE when added to 0134g indicates which location (0134g-0137g) is used for display update.

FIGURE 8-3A STATUS WORD LOCATION 0143g

0	1	2	3	4	5	6	7	8	9	10	11
SW ₀	X	X	X	X	X	SW ₆	SW ₇	X	X	X	X

X = Don't Care; SW₀ is MONITOR UPDATE flag

SW₀ = 0 - Display refreshed through user programmed DISPI-4 locations.

1 - USER PC used to update left display; right display to be determined by SW₆ and SW₇.

SW₆ = 1 - Right display contains memory data at User PC.

0 - SW₇ will determine right display.

SW₇ = 1 - Right display contains user accumulator.

0 - Right display contains word pointed to by 0127g.

FIGURE 8-3B SWITCH WORD LOCATION 0133g

OPTION I: OXX XXX XXX XXX in location 1133g

User loads location 0155g, 0156g and calls ODISLD or loads location 0134g, 0135g, 0136g, 0137g. CP interrupts will place the contents of 0155g and 0156g in the left and right displays, respectively.

OPTION II: 1XX XXX 1XX XXX in location 0133g

CP interrupts will place the User PC in the left display and data stored at the User PC in the right display.

OPTION III: 1XX XXX 01X XXX in location 0133g

CP interrupts will place User PC in the left display and AC in the right display.

OPTION IV: 1XX XXX 00X XXX in location 0133g

CP interrupts will place User PC in the left display and the contents of the location whose address is in location 0127g in the right display.

FIGURE 8-3C ACTIVE DISPLAY OPTIONS

The Octal Display Load routine ODISLD will place octal data passed through locations SAV2 and SAV3 (0155g, 0156g) into the four locations 0134g-0137g in the format shown in Figure 8-3C. BCD data may also be displayed but the four locations must then be loaded under user program control.

Example: Octal numbers ABCDg in 0155g; EFGHg in 0156g

A	B	C	D
---	---	---	---

E	F	G	H
---	---	---	---

A CALL to ODISLD will leave the DISP1-4 locations containing:

0134	0001	A	E	0135	0010	B	F
0136	0100	C	G	0137	1000	D	H

FIGURE 8-3D DISPLAY FORMATTING

The ODISLD routine makes use of a subroutine SHIFTY to shift digits. The shift count is passed to the subroutine as a constant following the CALL address. Thus the address of this constant is pushed onto the return address stack and the subroutine must access it via the stack, and increment the return address.

SWDB - ROM Locations 6200-6224, flow chart, Page 8-4

This routine reads the keypad into the accumulator, waits for several milliseconds, and again reads the keypad to see if it matches the first reading, thus indicating the end of switch bounce. If the readings do not match another timeout is allowed. During the timeout, the display is refreshed approximately every four milliseconds.

CLKPD - ROM locations 6156-6164, flow chart Page 8-4

This routine calls SWDB in order to timeout bounces, and checks for a zero reading from the keypad (indicating keypad clear) as long as required then returns to the calling program.

HEX - ROM locations 6441-6473, flow chart Page 8-4

This routine calls CLKPD to get a keypad clear indication, then this routine determines which key was pressed and generates a different number for each key. These numbers are used by the SHELL routine to generate starting addresses to the control state routines for each key.

EXIT - ROM locations 6051-6063, flow chart Page 8-2

This routine is entered when no keypad activity can be detected. The routine waits for the keypad to clear by executing CLKPD, then restores all registers and flags from RAM save locations. It then returns via the pointer in location 0000.

There is another entry point to this routine called OUT which is used if no keypad activity was detected even before key debouncing is needed, indicating the keypad was already clear. By entering at OUT, CLKPD does not have to be called, saving at least the 20 milliseconds it takes to execute SWDB.

CONTROL STATE SERVICE ROUTINES

Four of the control states possible through key depressions require extremely simple service routines. These four along with the symbolic starting address are:

INSPECT AC	INSAC
DECREMENT PC	DECPC
HALT	HALT
RUN	RUN

These routines are stored in ROM locations 6425-6440, and the flow charts are shown on Page 8-5.

These routines are each a few instructions long and self-explanatory. They modify the RAM save locations. INSAC complements bit 6 of the switch word in location 0133 (see Figure 8-3).

The control panel program when executing the EXIT routine restores all flags and registers in the IM6100 from these RAM save locations.

The RUN routine uses the IOT RUN, 6407, command described in Chapter 4.

Except for DECPC and INSAC, the above routines, when complete, branch to the EXIT routine described previously by jumping indirect via the location labeled UG. DECPC and INSAC, upon completion, jump indirect via BUG which is the starting address of SHELL, returning INTERCEPT JR. to the undefined control state. This enables the user to pick the next control state without again pressing the CNTRL key.

RESET, ROM locations 6165-6177, flow chart Page 8-4

A keypad RESET (CNTRL RESET) clears AC, FLAGS, MQ save locations, clears external device flags by pulling the microprocessor RESET line low during DEVSEL time (thus not affecting the microprocessor, which samples RESET during state time T1) and loads 0200 into SAVPC.

DEPOSIT INTO MEMORY, DEP, ROM locations 6502-6542, flow chart Page 8-5

This routine with starting address at DEP may be executed repeatedly when a sequence of numbers is entered from the keypad. It begins by calling the routine HEX. The value passed on by HEX is tested for being greater than 7. If it is not greater than 7, it is interpreted to be an octal digit to be deposited into memory by shifting it into the rightmost digit. This is done by getting the current memory data indirect via 0000g, SAVPC, shifting left three bits, while clearing the link each time so that zeros are shifted into the LSB, then adding the new digit. The routine PLACE is then called, which makes a range check and disallows writing into location 0000g (reserved for interrupt return addresses) or into locations 01438 - 0177g, as these locations are used by the MONITOR to store temporary variables.

If the digit is greater than 7, it is not to be entered into memory, but rather a pointer is computed to force a branch to the proper routine to be executed next. This is done by adding the contents of TAB, 6510g, to the value returned by HEX, 10, 11, 12, 13, resulting in 6520g, 6521g, 6522g,

65238. These locations contain pointers to routines DCI, PCI, EXIT and SHELL respectively.

In other words, pressing DECPC at this time results in routine DCI being executed, pressing MEM results in routine PCI being executed, pressing the yellow key results in the EXIT routine being executed and pressing the CNTRL key results in SHELL being executed, meaning a return to undefined control state.

Routine DCI decrements the PC by adding -1, 77778, to it, and returns to DEP to get the next digit, indicating the contents of the decremented memory location may now be altered.

Routine PCI increments the PC when key MEM is pressed and returns to DEP so that data may be entered into the incremented memory location.

These routines allow the user to step forwards and backwards through memory and alter data at will, as long as the memory area being addressed is not in ROM. ROM may be examined but not altered.

BLANK FLAG TOGGLE, BLK, ROM locations 6474-6501, flow chart Page 8-5

This routine is executed when the key marked DIS RAL ISZ is pressed when in the undefined control state. Bit #0 in the status word, Figure 8-3A, is called the blank flag, and this routine toggles it every time it is executed, therefore, allowing the user to shut off the display to conserve power and to turn it back on. The routine clears the AC and L, gets the status word, shifts bit #0 into the link (by doing a left shift), complements the link, shifts it back, restores status and goes to EXIT.

SET PROGRAM COUNTER, SETPC, ROM locations 6543-6573, flow chart Page 8-5

This routine, like DEP, accepts octal digits from the keypad. It begins by calling the routine HEX to get a valid number from a key depression. The value is checked for being over 7. If not, the routine goes on to GOON, which loads the digit into the rightmost octal position in the PC and jumps back to SETPC to pick up a new key depression.

If the value returned by HEX is greater than 7, a base address in location ADJT is added to it, and the sum is used as an indirect pointer back to SETPC (if the DECPC or MEM keys are pressed) to EXIT (if yellow key is pressed) or to SHELL (if CNTRL is pressed).

MICROINTERPRETER, MICRO, ROM locations 6600-7275, flow chart Page 8-6

Routine MICRO calls HEX and gets an index to compute a pointer to the routines servicing the individual keys (see Example 5 in Chapter 3 for a detailed description).

Pressing the IAC key causes AINC to be executed, incrementing SAVPC. Pressing any of the keys with memory reference instruction opcodes on them causes routines ATAD, AISZ, ADCA, AJMS or AJMP to be executed. These routines load the opcode into the AC and jump to AAND. (Note that the opcodes are sometimes stored as constants, and sometimes are instructions located elsewhere in the same page). After the opcode of a memory reference instruction (MRI) is interpreted, when the keypad is activated to enter an address digit, the value is first checked to be a valid digit (less than or equal to 7) and displayed as the least significant octal digit in the right display. When any numeric key is pressed, the opcode is shifted out and displayed continuously in the left display. The user can enter any string of octal digits into the right display from right to left, and terminate the string by a CNTRL keypress. If the absolute address in the right display is valid (page 0 address or current page address) the MICRO will interpret the instruction correctly along with the proper page bit magnitude. While the CNTRL key is depressed, MICRO will display the instruction on the right, the user PC on the left. As the CNTRL key is released, the left display increments and the MICRO mode is reentered for the next instruction.

If the IAC key is pressed without pressing any numeric key, the PC will increment and the MICRO mode will remain in effect. Note that the yellow IAC key is also labeled IND and may be used to set the indirect bit.

Routine MRPA continues to scan digits entered from the keypad and checks to see if they are address digits, 0-7, a CNTRL key depression (routine NEXT is executed in which the user PC is incremented, and control returns to MICRO to interpret the next instruction) or an IND key depression (in which case routine ZONK is entered in order to set indirect bit 3). This is done by rotating the indirect bit into the link, setting it and rotating back. Control is passed back to MRPA so it makes no difference if the indirect bit is set before or after the address bits are supplied.

MICRO, like SHELL, depends on MONITOR utility routines HEX, CLKPD, SWDB, PLACE, etc. in order to acquire valid keypad data and enter it into allowable memory space.

When interpreting MRI's, MICRO makes use of the different display mode options in the routine TOZE by loading SAV2 and SAV3 with the opcode and absolute address and calling ODISLD. MRPA is again entered to acquire the next digit while control panel interrupts cause the MONITOR to display the opcode and address. When address entry is terminated, routine ADTS checks if the address is in page 0 or in the current page (by comparing PC page bits with page bits of address) and either calls PLACE or branches to FLASH.

PLACE (Page 8-8, locations 7561-7577) makes a range check and disallows writing into location 0000g (reserved for interrupt return addresses) or into locations 0143g to 0177g as these locations are used by the MONITOR to store temporary variables.

If the absolute address is out of page, FLASH is entered, which flashes the display to indicate an invalid address field. The flash routine blanks the display using IOT instruction 6400 and times out approximately $(4096 \times (16 + 10) \times 10)$ or 1064960 states. This takes over half a second at 3.33 or 4 MHz.

FLASH then checks to see if the keypad has been depressed. If it has not, the routine continues to time out a different constant, TKB. If it has, the address field is cleared and subsequent depressions of the keys load the new digits in the address field.

Routine AIOT (Page 8-8, locations 7000-7042) is entered if in the MICRO mode, key IOT is pressed. An opcode of 6 is entered into the AC with a microprogrammed combination of Group I microinstructions and the routine collects digits from the keypad, while checking for a CNTRL key entry.

Detection of a CNTRL causes a branch to NEXT which increments SAVPC and returns to MICRO as before. Octal digits are shifted into the device address and control fields of the IOT instruction from right to left.

Routine AOPR1 (Page 8-9, locations 7043-7124) is entered when an operate group 1 instruction is to be loaded via the keypad. The routine stores 7000 into the user addressed location by calling PLACE with 7000 in the accumulator. Then HEX is called as further digits are expected.

A table of jump addresses is used as described in Example 5, Chapter 3 to branch to the proper routine.

The branches either cause the program to ignore the key and look for the next key depression, AOPR1 + 3, or call BSET11 or call an appropriate bit set subroutine, JA10-JA4. The bit set routines are used by routines in all three operate groups so they are coded as subroutines that may be nested in the MONITOR stack.

The bit set routines work by reading a constant, AAA-AAG (locations 7243-7251), corresponding to the appropriate bit being set into the AC, then jumping to the MBST routine. This routine stores the constant temporarily in MQ, clears the AC, gets the instruction in its current state, updates it by OR'ing in the MW, replaces it at the user addressed location by calling PLACE and returns.

This procedure is followed by all the operate group microinstruction service routines.

In other words, a table of jump addresses is used to computer a branch to either a bit set routine or back to the keypad reading sequence.

SINGLE INSTRUCTION EXECUTE, SIN, ROM locations 7400-7560, flow chart Page 8-11

This routine is useful in program development as a single instruction at a time may be executed allowing intermediate results to be examined under MONITOR control. This routine may only be used to single step through programs in RAM and not in ROM because software "breakpoints" are implemented by replacing the instruction at a breakpoint with a jump to the breakpoint processing subroutine and this requires writing into the memory.

SIN first initializes page 0 locations 0152 and 0153 labeled STORE and SHIFT to contain the instruction JMP I SHIFT and the address 7524. This initializes the breakpoint return linkage locations. Then it checks the instruction for a CALL (4161g), a JMP or a JMS. If it is none of these, it goes to the EXEC routine.

If it is a JMP or a JMS, the INAD routine is called to determine the next address to be accessed, this is placed in SAVPC for a pseudo-JMP and SHELL is reentered; to execute a pseudo-JMS, the current PC is incremented and stored at the next address (stored in TIME), the next address is incremented and replaces the contents of SAVPC, and SHELL is reentered.

Routines INAD and INDB determine whether the current page bit and indirect bit are set by masking off all other bits and testing for a non-zero AC. If the page bit is set, the current page number is obtained by masking off other bits. This page number is concatenated with the page address. If the indirect bit is set, the effective address is fetched and replaced in TIME. In any event, when location EXEC + 4 is reached, TIME contains the address of the next instruction to be fetched. Now the program gets the contents of this location, NEXT, and the next sequential one (NEXT + 1) and saves them in SAV1 and SAV2. The contents of these two locations are replaced by the instruction JMS BACK, which is 4151, a JMS to page 0 location 0151 and labeled BACK. Then both these locations are tested to see if the instruction was actually placed there, that is, if RAM exists there. The program does this by reading the locations back, adding the two's complement of 4151g to them and checking for a zero AC.

If the locations were indeed loaded correctly, the program proceeds to restore the MQ, LINK and AC and performs an indirect jump via SAVPC, executing the instruction specified by the user.

This instruction is executed, and, when the user program fetches the next instruction, it turns out to be the JMS BACK breakpoint placed by the MONITOR, so the user program stores the return address in BACK, 0151, and executes the instruction in location 0152 which happens to be the JMP I SHIFT which was placed there earlier. Thus, control is returned to the SIN routine at the point 7524 labeled RET. The routine saves away the AC, L and MQ again, restores the two instructions at the breakpoints, updates the user PC using the address stored in BACK and returns to the undefined control state.

The reason for storing JMS BACK in two successive locations can now be seen to provide for the case when the single instruction to be executed may skip the next location.

The MONITOR will allow all JMP, JMS, AND, IOT and OPERATE instructions including JMP*-1, JMS*-1, and JMS*-2 instructions to be single stepped properly. A limitation of the SIN program is that TAD, ISZ and DCA instructions which refer to a *+1 or *+2 location cannot be single stepped properly. There is little application for a program that uses instructions referencing the next sequential location, and especially, alters it, so we shall look at the cases when *+2 locations are accessed.

The instruction TAD*+2 will add the breakpoint instruction 4161 to the contents of the AC.

The instruction ISZ*+2 will increment the value 4161 to 4162 and then the original datum is restored so there is no net effect when single stepping this instruction.

The instruction DCA*+2 is useful in the INTERCEPT JR. to display a result when the location following this instruction contains the HALT instruction 7402. However, when single stepping this instruction, the DCA will write over the breakpoint instruction, then the original content is restored, so there is no net effect. It is recommended that the sequence

```
DCA*+3
NOP
HALT
```

is used to display data in programs when single stepping is desired.

A simpler alternative is to leave out the DCA instruction (so AC is not cleared) and select the Inspect AC mode before running the program. The right display will then show the AC.

PIE INITIALIZE, INPIE, PRINT TO TTY, TALK, RECEIVE FROM TTY KEYBOARD OR READER, LISN

These routines in ROM locations 6340-6362 and 7600-7621 are described in Chapter 7 on the PIEART board. See Page 8-12 for the flow chart.

INTERCEPT JR. BINARY LOADER, BIN, ROM locations 7622-7775, flow chart Page 8-13

This loader uses the PIEART interface board. The routine initializes the PIE-UART checksum and RAM locations it uses, then gets a character by calling LISN. The character is checked for being a rubout (all channels punched) or part of leader-trailer (only channel 8 punched), and if it is either, the program branches to RUM or LTC respectively. RUM continues to scan characters and echo all characters until another rubout is detected at which point it returns to BEG+1, which begins to process the next character. The system does not load text enclosed by rubouts.

LTC checks if the character is a first LT character or not. If so, the load routine is ended, the stray bit which appears on some PAL-8 generated tapes is masked, the checksum computed, the SAVAC location placed in the address display and the machine is halted showing the checksum.

If the character received was neither a rubout nor an LT character, the program updates the checksum, checks for a "change field" character (if it is, it is ignored and the next character is processed) and checks for "origin" data (if so, it gets the address data in two successive characters). The loader will ignore data for locations 0000g and 0143g-0177g. Data is loaded by routine DL2 only when conditions are valid.

INTERCEPT JR. MEMORY DUMP, DUMP, ROM locations 7305-7376, flow chart Page 8-14

This program requires that the first and last locations, of a block of memory to be dumped on tape, should be entered in locations 0131 and 0132, and the program run starting at location 7305.

The program uses leader-trailer routine TWTY contained in locations 6363-6374. It will punch out a BIN formatted tape complete with leader-trailer and checksum.

The program disables the CP request timer, initializes the PIE-UART, calls routine TWTY in the leader-trailer program to punch 63 LT characters.

The program next punches out the origin address, user entered in 0131, in two successive ASCII characters along with the channel 7 punch.

The data is also punched out using two characters per 12 bit word. The program counts the 1st and 2nd characters by looking at location BACK which is loaded with 7776 and incremented as a character is output. After two characters, the location becomes zero and the ISZ that incremented it will skip the BSW that is used to position the 2nd half of the character.

After every data item is transmitted, the address is checked to see if the end of the block has been reached.

As each character is punched (by calling the PUNCH routine, which in turn calls TALK), the checksum is updated in location SAV5.

After the last data item has been punched, the checksum is punched by CHSUM and routine TWTY is again called to punch out the leader-trailer tape.

Finally, the CP request timer is restored and the processor halted.

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 1		/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 2	
/ MONITOR 2 / THE MONITOR PROGRAM FOR THE INTERCEPT JR. / THIS PROGRAM RESIDES IN THE 17K212 ROM MARKED / "MICROINTERPRETER" AND IS ROM MASK 5504. THE / HAS BEEN DESIGNATED 15D006. / THE ROM HAS BEEN PROGRAMMED TO OCCUPY / ADDRESS SPACE 6000-7777 WITH THE RAM SELECT ACTIVE / FOR ADDRESS SPACE 0000-0177.		06023 1245 TAD BASE / YES! THIS IS A POWER-UP RESET 06024 3165 DCA STACY / RESET THE STACK POINTER 06025 7240 CLA CMA / SET THE AC 06026 3133 DCA SWITCH / SET THE SWITCH TO THE DEFAULT / CONDITION OF DISPLAY UPDATE ON / AND MD (MEMORY DATA) DISPLAY ON 06027 4161 CALL / REFRESH THE DISPLAY REF5H 06031 7604 LAS / LOAD THE KEYPAD TO THE AC 06032 7500 SNA / LOOK FOR A "C" KEYPRESS 06033 5233 JMP OUT / NO! GO TO OUT 06034 4161 CALL / YES! TEST TO SEE IF IT IS A VALID 06035 6200 SWDB / SWITCH PRESS. 06036 7700 SNA CLA / NO! GO TO EXIT 06037 5251 JMP EXIT / YES! GO TO THE PROGRAM SHELL 06040 5661 JMP I,+1 06041 4400 SHEL	
6000 START=6000 / THE PAGE ZERO VARIABLES FOR THE PROGRAM / THE MONITOR RESERVES LOCATIONS 130-177 / FOR ITS OWN USE AND RESTRICTS THE USER / ACCESS TO THESE LOCATIONS. / THESE LOCATIONS ARE ACCESSIBLE TO THE USER / HOWEVER THEY MAY BE DISTURBED BY THE MONITOR		06042 5563 JMP1. / THE TABLE OF CONSTANTS 06043 6064 KCALLY. CALLY 06044 6075 KRETY. RETY 06045 0167 BASE. / STACK+2 06046 5777 RAM2L. / KEYHOLE POINTER 06047 0764 RAM2R. / KEY 06050 4000 RAM2S. / 4000	
#0130 #130 00130 0000 HOLD1. 0 00131 0000 HOLD2. 0 00132 0000 HOLD3. 0 00133 0000 SWITCH. 0 00134 0000 DISP1. 0 00135 0000 DISP2. 0 00136 0000 DISP3. 0 00137 0000 DISP4. 0 / THE FOLLOWING LOCATIONS CANNOT BE LOADED / USING THE BIN LOADER OR THE MEMORY DEPOSIT / OR MICRO ROUTINES IN THE MONITOR #0000 #0000 06000 0000 SAVPC. 0000 #0140 #0140 00140 0000 SAVAC. 0		06051 4161 EXIT. CALL / WAIT FOR THE KEYPAD TO CLEAR 06052 6156 CLKPD / CLEAR THE AC AND LHM 06053 7300 OUT. CLA CLL / RESTORE THE MD TO THE USER 06054 1142 TAD SAVPC / RESTORE THE LHM TO THE USER 06055 7421 MD. CLA 06056 1141 TAD SAVL / CLEAR THE AC 06057 7104 RAL. CLA 06060 7200 CLA / RESTORE THE AC TO THE USER 06061 1140 TAD SAVPC / RESTORE THE INTERRUPT ENABLE FLAG TO THE 06062 6001 TON. / USER AND COME OUT OF CP MODE 06063 5400 JMP I SAVPC / RESTORE THE MD TO THE USER	
/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 1-1		/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 2-1	
00141 0000 SAVFL. 0 00142 0000 SAVMD. 0 00143 0000 STATUS. 0 00144 0000 TIME. 0 00145 0000 SAVE. 0 00146 0000 HOLD. 0 00147 0000 POINT. 0 00150 0000 TEMP. 0 00151 0000 BACK. 0 00152 0000 STORE. 0 00153 0000 SHFT. 0 00154 0000 SAV1. 0 00155 0000 SAV2. 0 00156 0000 SAV3. 0 00157 0000 SAV4. 0 00160 0000 SAV5. 0 / THE PAGE ZERO LOCATIONS FOR THE MONITOR / STACK #0161 #161 00161 0000 CALLX. 0 #0164 #164 00164 0000 RETL. 0 00165 0000 STAC. 0 00166 0000 AC. 0		/ THE EXIT ROUTINES FOR THE MONITOR 06064 3166 CALLY. / DCA AC 06065 2165 IZT STAC / UPDATE THE STACK POINTER 06066 1161 TAD CALLX / CALLY HAS THE RETURN ADDRESS 06067 7001 INC. / INCREMENT THE RETURN ADDRESS 06070 3565 DCA I STAC / SAVE ON THE LIFO STACK 06071 1561 TAD I CALLX / GET THE USER SUBROUTINE 06072 3161 DCA CALLY / PUT IT ON CALLY 06073 1166 TAD AC / RESTORE THE AC 06074 5561 JMP I CALLX / GO TO THE SUBROUTINE CALLED 06075 3166 RETY. / DCA AC 06076 1565 TAD I STAC / GET THE RETURN ADDRESS FROM THE STACK 06077 3161 DCA CALLX / AND PUT IT IN CALLY 06100 7040 CMA CML / COMPLEMENT THE AC AND LHM 06101 1165 TAD STAC / DECREMENT THE STACK AND RESTORE THE 06102 3165 DCA STAC / USER LHM 06103 1166 TAD AC / RESTORE THE STACK POINTER 06104 5561 JMP I CALLX / RETURN TO THE PROGRAM	
/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 1-2		/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 2-2	
#7777 #7777 07777 5776 JMP I,+1 #7776 #7776 07776 6000 INIT / THE PROGRAM START POINTER / THE INIT ROUTINE FOR THE MONITOR / PERFORMS THE ELEMENTARY FUNCTIONS / OF THE MONITOR. #6000 #START 06000 3140 INIT. / DCA SAVAC / SAVE THE USER AC IN RAM 06001 6004 GTR / GET THE USER FLAGS 06002 3141 DCA SAVFL / SAVE THE USER FLAGS IN RAM 06003 7521 SWP / GET THE USER MD AND CLEAR THE MD 06004 3142 DCA SAVMD / SAVE THE USER MD IN RAM 06005 1242 TAD JMP1. / ESTABLISH THE CP SUBROUTINE STACK 06006 3162 DCA CALLX+1 / LOCATIONS IN RAM 06007 1243 TAD KCALLY 06010 3163 DCA CALLX+2 06011 1244 TAD KRETY 06012 3164 DCA RETX / THE LINK TO THE ROM ADDITION / THE PLACEMENT OF A KEY IN THE / KEYPAD OF THE ROM ADDITION LOCATION / 5777 WHICH MATCHES THE KEY IN THE / MONITOR PROGRAM VALUE # 7014 / WILL CAUSE THE CONTROL PROGRAM / TO BRANCH TO THE ROM ADDITION / AT THIS POINT. 06013 1646 TAD I RAM2L 06014 1247 TAD RAM2C 06015 7650 SNA CLA 06016 5650 JMP I RAM2J / THE MONITOR NOW BRANCHES / TO THE ROM ADDITION 06017 1000 TAD SAVPC / GET THE USER PC 06020 7001 INC / INCREMENT THE VALUE 06021 7640 SZA CLA / TEST: WAS THE USER PC EQUAL TO 7777 06022 5227 JMP,+5 / NO! THIS IS NOT A POWER-UP CP CALL		/ THE SUBROUTINE STACK FROM OVERHEAD 06064 3166 CALLY. / DCA AC 06065 2165 IZT STAC / UPDATE THE STACK POINTER 06066 1161 TAD CALLX / CALLY HAS THE RETURN ADDRESS 06067 7001 INC. / INCREMENT THE RETURN ADDRESS 06070 3565 DCA I STAC / SAVE ON THE LIFO STACK 06071 1561 TAD I CALLX / GET THE USER SUBROUTINE 06072 3161 DCA CALLY / PUT IT ON CALLY 06073 1166 TAD AC / RESTORE THE AC 06074 5561 JMP I CALLX / GO TO THE SUBROUTINE CALLED 06075 3166 RETY. / DCA AC 06076 1565 TAD I STAC / GET THE RETURN ADDRESS FROM THE STACK 06077 3161 DCA CALLX / AND PUT IT IN CALLY 06100 7040 CMA CML / COMPLEMENT THE AC AND LHM 06101 1165 TAD STAC / DECREMENT THE STACK AND RESTORE THE 06102 3165 DCA STAC / USER LHM 06103 1166 TAD AC / RESTORE THE STACK POINTER 06104 5561 JMP I CALLX / RETURN TO THE PROGRAM / WE NOW CONTINUE WITH THE MONITOR / SUBROUTINE: / THE DISPLAY REFRESH ROUTINE / THE AC AND THE LHM ARE LOST 06105 7300 REF5H. / CLA CLL 06106 1143 TAD STATUS / GET STATUS 06107 7710 SPA CLA / TEST: IS THE DISPLAY FLAG SET? 06110 5314 JMP,+4 / YES! GO ON 06111 8400 THRU. / 8400 / NO! COME OUT OF THE ROUTINE 06112 7200 CLA CLL / CLEAR THE AC AND THE LHM 06113 5564 RETURN / RETURN TO THE PROGRAM 06114 1133 TAD SWITCH / GET THE SOFTWARE SWITCH 06115 7700 SNA CLA / TEST: IS THE MONITOR UPDATE FLAG SET? 06116 5334 JMP UDIS / NO! REFRESH THE USER DISPLAY 06117 1000 TAD SAVPC / YES! GET THE USER PC 06120 3155 DCA SAV2 / STORE IN SAV2 TO PASS TO ODISLD 06121 1133 TAD SWITCH / GET THE SOFTWARE SWITCH 06122 7002 BSW / POSITION THE OTHER FLAGS 06123 7610 DCA MDIS / TEST: IS THE AC DATA FLAG SET? 06124 5251 JMP MDIS / YES! GO GET THE MEMORY DATA 06125 7604 RAL / NO! POSITION BIT # 06126 7710 SNA CLA / TEST: IS THE AC DISPLAY FLAG SET? 06127 5254 JMP ACDIS / YES! GO GET THE USER AC 06130 1527 TAD I 127 / NO! GET THE WORD POINTED TO BY 127 06131 3156 DR. / DCA SAV2 THRU SAV3 06132 4161 CALL / UPDATE DISP1-DISP4 06133 6225 ODISLD 06134 1143 UDIS. / TAD STATUS / GET THE STATUS WORD 06135 0247 AND MSB1 / CLEAR THE BIT BUCKET 06136 3143 DCA STATUS / RESTORE STATUS 06137 1143 TAD STATUS / GET STATUS 06140 0250 AND MSB2 / MASK OUT THE DIGIT CODE 06141 1346 TAD TABJ / ADJUST TO THE TABLE OF 06142 3147 DCA POINT / DISPLAY WORDS 06143 1547 TAD I POINT / PLACE AT A POINTER 06144 2143 IZT STACY / RESTORE THE AC 06145 5311 JMP THRU / INCREMENT THE DIGIT CODE / GO LOAD THE DISPLAY 06146 0134 TADJ. / DISP1 06147 7773 MSB1. / 7773 06150 0003 MSB2. / 0003 / THE TABLE OF CONSTANTS 06151 7300 MDIS. / CLA CLL 06152 1400 TAD I SAVPC / CLEAR THE AC AND LHM 06153 5031 JMP ON / GET THE MEMORY DATA / GO UPDATE 06154 1140 ACDIS. / TAD SAVAC / GET THE USER AC 06155 5031 JMP ON / GO UPDATE / THE CLEAR KEYPAD ROUTINE / THE AC AND LHM ARE NOT AFFECTED 06156 3154 CLYPD. / DCA SAV1 / SAVE THE AC IN SAV1 06157 4161 CALL / GET A SWITCH READING 06160 6200 SWDB / GET A SWITCH READING 06161 7440 SZA / TEST FOR A ZERO READING 06162 4161 JMP,+5 / NO! GO BACK AND TRY AGAIN 06163 1174 SNA SAV1 / YES! RESTORE THE AC 06164 5514 RETURN / RETURN TO THE PROGRAM / THE RESET ROUTINE	

INTERFIL

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 2-3

```

06145 7340 RESET. CLA CLL CMA          / SET THE STATUS WORD
06146 3143 DCA STATUS          / CLEAR THE AC, LIM, FLAGS, AND MC
06147 3140 REFSAVAC   / FOR THE USER, SET THE STATUS
06170 3141 DCA SAVFL      / WORD SO THAT THE DISPLAY IS ON
06171 3142 DCA SAVWD      /
06172 4406 64C        /
06173 1276 TAD CRUMP   / SET THE USER PC TO 200
06174 3000 DCA SAVPC  /
06175 3777 JMP I FAD        / GO TO HALT

06176 0200 CRUMP. 0200 / THE TABLE OF CONSTANTS
06177 4434 FAD. HALT
    
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 4-1

```

06302 3150 DCA TEMP          / PLACE IN TEMP
06303 1155 TAD SAV2          / GET THE FIRST DISPLAY WORD
06304 0321 AND MSC4          / MASH OUT THE LAST DIGIT
06305 7002 BSH           / POSITION
06306 4161 CALL          / SHIFT OVER 2 PLACES
06307 4322 SHIFTY          /
06310 7776          /
06311 1156 TAD SAV3          / GET THE SECOND DISPLAY WORD
06312 0321 AND MSC4          / MASH OUT THE LAST DIGIT
06313 1150 TAD TEMP          / COMBINE WITH TEMP
06314 3137 DCA DISPA        / PLACE IN DISPA
06315 3564 RETURN         / GO BACK TO THE PROGRAM

06316 7000 MSC1. 7000 / THE MASH TABLE
06317 0700 MSC2. 0700
06320 0070 MSC3. 0070
06321 0007 MSC4. 0007

06322 3146 SHIFTY. DCA HOLD / STORE THE DIGIT IN SAVE
06323 1565 TAD I STACK / GET THE POINTER TO THE SHIFT COUNT
06324 3147 DCA POINT / PLACE IN POINT
06325 1547 TAD I POINT / GET THE SHIFT COUNT
06326 3153 DCA SHIFT / PLACE IN SHIFT
06327 2545 BUMP I STACK / BUMP THE RETURN ADDRESS BY ONE
06330 1146 TAD HOLD / GET THE DIGIT
06331 7110 CLL RAR / ROTATE IT ONE PLACE RIGHT
06332 2193 ISZ SHIFT / INCREMENT THE COUNT
06333 5331 JMP I 2 / NOT DONE SHIFTING YET
06334 1156 TAD TEMP / ALL DONE, NOW COMBINE WITH TEMP
06335 3150 DCA TEMP /
06336 3564 RETURN / GO BACK TO THE PROGRAM
    
```

/ REFER TO THE LATER PART OF THIS LISTING
 / FOR THE INFC ROUTINE SOURCE WHICH IS
 / LOCATED AT THIS POINT IN THE ADDRESS
 / SPACE.

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 3

 / CROSS THE PAGE BOUNDARY TO PAGE #2

*6200 *START=200

/ THE SWITCH DEBOUNCE ROUTINE
 / LOSE THE AC AND THE LIM

```

06200 7604 SHDB. LAS          / READ THE KEYPAD INTO THE AC
06201 3145 DCA SAVE          / STORE IN SAVE
06202 1224 TAD TCNT          / GET THE WAIT COUNT
06203 3157 DCA SAV4          / PLACE IN THE COUNTER
06204 1223 TAC TX1         / GET THE TIME CONSTANT
06205 3144 DCA TIME         / PLACE IN THE TIMER
06206 2144 ISZ TIME         / TIME OUT 5 MILLISECONDS
06207 5206 JMP I -1 / REFRESH THE DISPLAY
06210 4161 CALL          /
06211 8105 REFSAVAC   / COUNT DOWN THE WAIT COUNT
06212 2157 ISZ SAV4        /
06213 5204 JMP I -7 / GET ANOTHER SWITCH READING
06214 7604 SHDB.          / NEGATE IT
06215 7041 CIA           / ADD IN THE FIRST READING
06216 1145 TAD SAVE          / TEST FOR A MATCH
06217 7640 SZA CLA        / NO MATCH SO DO IT AGAIN
06220 5200 JMP SWDB          /
06221 1145 TAD SAVE          / THERE IS A MATCH SO GET THE
06222 5564 RETURN         / KEYPAD READING INTO THE AC
                                / GO BACK TO THE PROGRAM

06223 7620 TX1. 7620 / THE TABLE OF CONSTANTS
06224 7776 TCNT. 7776
    
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 5

 / CROSS THE PAGE BOUNDARY TO PAGE #3

*6400 *START=400

THE MONITOR SHELL PROGRAM

```

06400 4161 SMELL. CALL / GET A NUMBER FROM THE KEYPAD
06401 4481 HEX          /
06402 1207 TAD GOTO        / ADJUST A POINTER TO THE TABLE
06403 3147 DCA POINT / PLACE IN POINT
06404 1547 TAD I POINT / GET THE ROUTINE ADDRESS
06405 3147 DCA POINT / PLACE IN POINT
06406 5547 JMP I POINT / GO TO THE ROUTINE

06407 4410 GOTO. GOTO+3 / ADJUSTMENT VALUE FOR THE TABLE
06410 8600 MICRO / THE TABLE OF POINTERS
06411 7622 EIM          /
06412 6474 ELI          /
06413 7400 SIN          /
06414 6434 RUN          /
06415 6434 HALT         /
06416 6165 RESET        /
06417 6565 SETPC       /
06420 6430 DEPCPC      /
06421 6502 DEP          /
06422 4430 INAC        /
06423 6400 BUG. SMELL /
06424 6071 UG. EXIT
    
```

THE INSPECT AC ROUTINE

```

06425 4161 INSA. CALL / GO TO AC FLAG TOGGLE
06426 7276 RINSAC /
06427 5623 JMP I BUG / GO TO THE SMELL
    
```

THE DECREMENT PC ROUTINE

```

06430 7540 DEPCPC. CLA CLL CMA / SET THE AC TO -1
06431 1000 TAD SAV4 / ADD THE DECF #1
06432 3000 DCA SAVPC / RESTORE THE DECREMENTED PC
06433 5623 JMP I BUG / GO TO THE SMELL
    
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 4

/ THE OCTAL DISPLAY LOAD ROUTINE
 / ARGUMENTS ARE PASSED THRU SAV2 AND
 / SAV3. SAV2 WILL GO TO DISPLAY #1
 / AND SAV3 WILL GO TO DISPLAY #2

```

06225 7307 ODISLD. CLA CLL IAC RTR / SET THE AC TO 0004
06226 7002 BSH           / SET THE AC TO 0400
06227 3130 DCA TEMP          / PLACE IN TEMP
06230 1155 TAD SAV2          / GET THE FIRST DISPLAY WORD
06231 0314 AND MSC1          / MASH OUT THE FIRST DIGIT
06232 4161 CALL          / SHIFT OVER 5 PLACES TO THE RIGHT
06233 6322 SHIFTY          /
06234 7773          /
06235 1156 TAD SAV3          / GET THE SECOND DISPLAY WORD
06236 0316 AND MSC1          / MASH OUT THE FIRST DIGIT
06237 4161 CALL          / SHIFT OVER 5 PLACES TO THE RIGHT
06240 6322 SHIFTY          /
06241 7767          /
06242 1150 TAD TEMP          / PLACE THE COMPLETE DISPLAY WORD
06243 3134 DCA DISP1        / INTO DISP1

06244 7332 CLA CLL CML RTR / SET THE AC TO 1000
06245 7010 RAR           /
06246 3150 DCA TEMP          / PLACE IN TEMP
06247 1155 TAD SAV1          / GET THE FIRST DISPLAY WORD
06250 0317 AND MSC2          / MASH OUT THE SECOND DIGIT
06251 4161 CALL          / SHIFT OVER 2 PLACES
06252 6322 SHIFTY          /
06253 7776          /
06254 1156 TAD SAV3          / GET THE SECOND DISPLAY WORD
06255 0317 AND MSC2          / MASH OUT THE SECOND DIGIT
06256 4161 CALL          / SHIFT OVER 6 PLACES
06257 6322 SHIFTY          /
06260 7772          /
06261 1150 TAD TEMP          / PLACE THE DISPLAY WORD
06262 3135 DCA DISP2        / INTO DISP2

06263 7332 CLA CLL CML RTR / SET THE AC TO 2000
06264 3150 DCA TEMP          / PLACE IN TEMP
06265 1155 TAD SAV2          / GET THE FIRST DISPLAY WORD
06266 0320 AND MSC3          / MASH OUT THE THIRD DIGIT
06267 7104 CALL          / SHIFT LEFT ONCE
06270 1150 TAD TEMP          / COMBINE WITH TEMP
06271 3130 DCA TEMP          /
06272 1156 TAD SAV3          / GET THE SECOND DISPLAY WORD
06273 0320 AND MSC3          / MASH OUT THE THIRD DIGIT
06274 4161 CALL          / SHIFT OVER THREE PLACES
06275 6322 SHIFTY          /
06276 7775          /
06277 1150 TAD TEMP          / GET THE DISPLAY WORD
06300 3136 AND DISP3        / AND PLACE IN DISP3

06301 7930 CLA CLL CML RAR / SET THE AC TO 4000
    
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 5-1

THE HALT ROUTINE

```

06424 7402 HALT. HLT / CLEAR THE RUN FLIP-FLOP
06425 5624 JMP I UG / GO TO EXIT
    
```

THE RUN ROUTINE

```

06436 7402 RUN. HLT / CLEAR THE RUN FLIP-FLOP
06437 6407 A437 / NOT RUN COMMAND
06440 5624 JMP I UG / GO TO EXIT
    
```

INTER-SIL

/ MONITOR 2 IFD05 PAL 1A 06-APR-77 PAGE 6

```

06441 7300 HEX. CLA CLL
06442 4161 CALL
06443 4156 CLKPD
06444 4161 HEXUC. CALL
06445 6200 SWDP
06446 7450 SRA
06447 5244 JMP HEXUC
06450 7010 RAR
06451 7430 SZL
06452 5255 JMP -*3
06453 2154 ISZ SAV1
06454 5250 JMP -4
06455 7300 CLA CLL
06456 1154 TAD SAV1
06457 1272 TAD KMP7
06460 7440 SZA
06461 5265 JMP -*4
06462 7201 CTA IAC
06463 1154 OK2. TAD SAV1
06464 5264 RETURN
06465 1273 TAD KMB
06466 7640 SZA CLA
06467 5263 JMP OK2
06470 7260 CLA CMA CML
06471 5263 JMP OK2
06472 7771 KMP7. 7771
06473 7777 KMB. 7777
    
```

/ THE HEX DIGIT ROUTINE
 / THIS ROUTINE TAKES A KEYPRESS
 / FROM THE KEYPAD AND CONVERTS IT
 / TO A HEX DIGIT FROM 0 TO
 / 11 IN THE AC.

/ CLEAR THE AC AND THE LINK
 / WAIT FOR A CLEAR KEYPAD
 / GET A READING FROM THE KEYPAD
 / TEST FOR A KEYPRESS
 / NO! GO BACK
 / YES: ROTATE RIGHT ONCE
 / TEST! WAS THAT BIT SET
 / YES! GO TO THE END
 / NO! INCREMENT THE ROTATE COUNT
 / GO BACK AND ROTATE AGAIN
 / CLEAR THE AC AND THE LINK
 / GET THE HEX NUMBER
 / SUBTRACT 7
 / TEST! IS IT EQUAL TO 7
 / NO! GO ON
 / YES INCREMENT THE VALUE TO 8
 / GET THE FINAL VALUE INTO THE AC
 / GO BACK TO THE PROGRAM

/ TEST! WAS THE VALUE 8?
 / SZA CLA
 / NO! ON TO RETURN
 / YES! SET THE AC TO -1
 / DECREMENT THE VALUE TO 7
 / -8

/ THE BLANK FLAG TOGGLE ROUTINE
 / THE AC AND LINK ARE LOST

```

06474 7300 BLK. CLA CLL
06475 1143 TAD STATUS
06476 7004 RAL
06477 7030 CML RAR
06480 1143 DCA STATUS
06501 5624 JMP I UG
    
```

/ CLEAR THE AC AND THE LINK
 / GET THE STATUS WORD
 / PLACE THE FLAG IN THE LINK
 / TOGGLE THE FLAG AND RESTORE
 / RESTORE STATUS WORD
 / GO TO EXIT

/ MONITOR 2 IFD05 PAL 1A 06-APR-77 PAGE 6-1

/ THE MEMORY DEPOSIT ROUTINE

```

06502 4161 DEP. CALL
06503 4441 HEX
06504 3150 DCA TEMP
06505 1150 TAD TEMP
06506 0307 AND SNERD
06507 7650 SNERD. SNA CLA
06510 5324 JMP PON
06511 1150 TAD TEMP
06512 1317 TAD TAB
06513 3150 DCA TEMP
06514 1350 TAD I TEMP
06515 3150 DCA TEMP
06516 5550 JMP I TEMP
06517 6510 TAB. TAB-7
06520 6524 DCA I
06521 6540 PCI
06522 6051 EXIT
06523 6400 SHEL
06524 1400 PON. TAD I SAVPC
06525 7104 CLL RAL
06526 7104 CLL RAL
06527 7104 CLL RAL
06530 1150 TAD TEMP
06531 4161 CALL
06532 7561 PLACE
06533 5302 JMP DEF
06534 7360 DCI. CLA CLL CMA CMA
06535 1090 JMP SAVPC
06536 3090 DCA SAVPC
06537 5302 JMP DEF
06540 2090 PCI. ISZ SAVPC
06541 7090 NOP
06542 5302 JMP DEF
    
```

/ GET A HEX VALUE FROM THE KEYPAD
 / PLACE IT IN TEMP
 / GET THE VALUE FROM TEMP
 / MASK OUT BIT #6
 / TEST! IS THE VALUE 7
 / NO! GO TO LOAD THE MEMORY
 / YES! GET THE VALUE
 / ADJUST TO POINT AT THE TABLE
 / PLACE IN TEMP
 / GET THE POINTER
 / PLACE IN TEMP
 / GO TO THE ROUTINE
 / THE TABLE OF POINTERS
 / DECREMENT THE USER PC
 / RESTORE THE PC
 / GO GET THE NEXT DIGIT
 / INCREMENT THE USER PC
 / IN CASE THE PC WAS 7777
 / GO GET THE NEXT DIGIT

/ MONITOR 2 IFD05 PAL 1A 06-APR-77 PAGE 7

/ THE SETPC ROUTINE
 / THE AC AND LINK ARE LOST

```

06543 4161 SETPC. CALL
06544 4441 HEX
06545 3150 TAD TEMP
06546 1150 TAD TEMP
06547 0307 AND MASH
06550 7650 MASH. SNA CLA
06551 5360 JMP OOCN
06552 1150 TAD TEMP
06553 1347 TAD ADIT
06554 3150 DCA TEMP
06555 1350 TAD I TEMP
06556 3150 DCA TEMP
06557 5550 JMP I TEMP
06560 1000 OOCN. TAD SAVPC
06561 7104 CLL RAL
06562 7104 CLL RAL
06563 7104 CLL RAL
06564 1150 TAD TEMP
06565 3090 DCA SAVPC
06566 5343 JMP SETPC
06567 6560 ADJT. ADJT-7
06570 6543 SETA
06571 6543 SETPC
06572 6051 EXIT
06573 6400 SURF. SHEL
    
```

/ GET A HEX NUMBER FROM THE KEYPAD
 / STORE IN TEMP
 / GET THE VALUE FROM TEMP
 / MASK OUT BIT #6
 / TEST! IS THE VALUE 7
 / NO! GO TO LOAD THE PC
 / YES! GET THE VALUE
 / ADJUST TO POINT AT THE TABLE
 / PLACE IN TEMP
 / GET THE POINTER FROM THE TABLE
 / PLACE THE POINTER IN TEMP
 / GO TO THE PROPER ROUTINE
 / GET THE USER PC
 / SHIFT IT OVER ONE OCTAL DIGIT
 / PLACE IN THE USER PC LOCATION
 / GO GET THE NEXT DIGIT
 / THE TABLE OF POINTERS

/ CROSS THE PAGE BOUNDARY TO PAGE #4

*6600 *START*600.

/ THIS IS THE MICROINTERPRETER
 / PROGRAM WHICH IS ENTERED FROM THE
 / MONITOR BY DEPRESSING THE "MICRO"
 / KEY FOLLOWING "READY" OF CONTROL
 / KEY PRESS. THE PROGRAM IS EXITED BY
 / TWO CONTROL KEYPRESSES IN SUCCESSION

/ MONITOR 2 IFD05 PAL 1A 06-APR-77 PAGE 7-1

```

06600 4161 MICRO. CALL
06601 4441 HEX
06602 1207 TAD XEO
06603 3147 DCA POINT
06604 1547 TAD I POINT
06605 3147 DCA POINT
06606 5247 JMP I POINT
06607 6610 XEO. XEO-1
06610 6643 AND
06611 6627 ATAD
06612 6631 RI3Z
06613 6633 ADCA
06614 6636 ALMP
06615 6640 ALMP
06616 7000 ADIT
06617 7125 ADPR1
06620 7043 ADPR1
06621 7202 ADPR2
06622 6624 ATNC
06623 6400 SHEL
06624 2000 AINC. ISZ SAVPC
06625 5200 JMP MICRO
06626 5200 JMP MICRO
06627 1313 ATAD. TAD K1000
06630 5243 JMP AAND
06631 7332 A15Z. CLA CLL CMA RTR
06632 5243 JMP AAND
06633 1235 ADCA. TAD I 2000
06634 5243 JMP AAND
06635 3000 I3000. 3000
06636 7313 ALMS. CLA CLL IAC RTR
06637 5243 JMP AAND
06640 1242 ALMP. TAD I 5000
06641 5243 JMP AAND
06642 5000 I5000. 5000
06643 4161 AAND. CALL
06644 7561 PLACE
06645 3161 DCA SAVS
06646 4161 MRPA. CALL
06647 4441 HEX
06648 1150 TAD TEMP
06649 1150 TAD TEMP
06650 0307 AND I01
06651 7650 SNERD. SNA CLA
06652 5360 JMP TOZ
06653 1150 TAD TEMP
06654 1150 TAD TEMP
06655 1150 TAD TEMP
    
```

/ GET A HEX VALUE FROM THE KEYPAD
 / ADJUST TO POINT AT THE TABLE
 / PLACE IN THE POINTER
 / GET THE JUMP ADDRESS
 / PLACE IN POINT
 / GO TO THE PROPER ROUTINE
 / THE TABLE OF POINTERS
 / INCREMENT THE USER PC
 / RETURN FOR NEW MICRO COMMAND
 / SET THE AC TO 1000
 / GO TO MRPA
 / SET THE AC TO 2000
 / GO TO MRPA
 / SET THE AC TO 3000
 / GO TO MRPA
 / SET THE AC TO 4000
 / GO TO MRPA
 / SET THE AC TO 5000
 / GO TO MRPA
 / PLACE THE CODE IN THE
 / MEMORY LOCATION
 / CLEAR THE RESOLUTION ADDRESS
 / GET A HEX VALUE FROM THE KEYPAD
 / STORE IN TEMP
 / GET THE VALUE FROM TEMP
 / MASK OUT BIT #6
 / TEST! IS THE VALUE 7
 / NO! GO TO ADDRESS LOAD
 / YES! ADJUST TO TABLE
 / ADD THE CHARACTER

/ MONITOR 2 IFD05 PAL 1A 06-APR-77 PAGE 7-2

```

06657 3130 DCA TEMP
06660 1530 TAD I TEMP
06661 3130 DCA TEMP
06662 5550 JMP I TEMP
06663 6654 ELO. ELO-7
06664 6646 MRPA
06665 6646 MRPA
06666 6735 IZMP
06667 6706 ADTS
06670 1160 TOZ. TAD SAVS
06671 7104 CLL RAL
06672 7104 CLL RAL
06673 7104 CLL RAL
06674 1150 TAD TEMP
06675 3160 DCA SAVS
06676 3133 TOZE. DCA SWITCH
06677 1400 TAD I SAVPC
06700 3135 TAD SAVS
06701 1160 TAD SAVS
06702 3156 DCA SAV2
06703 4161 DCA SAV3
06704 6225 OD15LD
06705 5246 JMP MRPA
06706 1160 ADTS. TAD SAVS
06707 0251 AND IUG1
06710 7450 SNA
06711 5323 JMP PUMP
    
```

/ PLACE IN TEMP
 / GET THE POINTER
 / PLACE IN TEMP
 / GO TO THE PROPER ROUTINE
 / THE TABLE OF POINTERS
 / GET THE ABSOLUTE ADDRESS
 / ROTATE IT OVER ONE OCTAL
 / DIGIT
 / ADD IN THE DIGIT
 / PLACE IN THE ABSOLUTE ADDRESS
 / CLEAR THE DISPLAY SOFTWARE
 / SWITCH
 / GET THE INSTRUCTION SO FAR
 / PLACE IN DISPLAY #1
 / GET THE ABSOLUTE ADDRESS
 / PLACE IN THE SECOND DISPLAY
 / LOAD THE DISPLAY MEMORY LOCATION
 / GO GET THE NEXT DIGIT
 / TAKE THE ABSOLUTE ADDRESS
 / MASK OUT THE PAGE ADDRESS
 / TEST! IS IT PAGE ZERO
 / YES! ON TO PLACE IN MEMORY

/ MONITOR 2 IFD05 PAL 1A 06-APR-77 PAGE 8

```

06712 3146 DCA HOLD
06713 1000 K1000. TAD SAVPC
06714 0251 AND IUG1
06715 7041 CIA
06716 1146 TAD HOLD
06717 7640 SZA CLA
06720 3352 JMP FLASH
06721 4161 CALL
06722 7252 BSETA
06723 1160 TAD SAVS
06724 0350 AND IUG1
06725 1400 TAD I SAVPC
06726 4161 CALL
06727 7561 PLACE
06730 7340 CLA CLL CMA
06731 3133 DCA SWITCH
06732 4161 CALL
06733 6156 CLKPD
06734 5345 JMP NEXT
06735 1400 ZONV. TAD I SAVPC
06736 7106 CLL RTL
06737 7026 RTL
06740 7132 CLL CMA RTR
06741 7012 RTR
06742 4161 CALL
06743 7561 PLACE
06744 5276 JMP TOZE
06745 2000 NEXT. ISZ SAVPC
06746 5200 JMP MICRO
06747 5200 JMP MICRO
06750 0177 TUG. 0177
06751 7600 TUG1. 7600
    
```

/ STORE IN HOLD
 / GET THE MEMORY ADDRESS
 / MASK OUT THE PAGE NUMBER
 / NEGATE IT
 / ADD IN THE PAGE NUMBER OF THE
 / ABSOLUTE ADDRESS
 / TEST! ARE WE IN CURRENT PAGE
 / NO! FLASH AN ERROR CONDITION
 / YES! SET THE CURRENT PAGE BIT
 / GET THE ABSOLUTE ADDRESS
 / MASK OUT THE PAGE ADDRESS
 / COMBINE WITH THE OP CODE IN
 / THE INSTRUCTION LOCATION
 / PLACE IN THE MEMORY
 / SET THE AC
 / SET THE SOFTWARE SWITCH TO
 / WAIT FOR THE END OF THE KEYPRESS
 / THE ASSEMBLED INSTRUCTION
 / GO GET THE NEXT MICRO COMMAND
 / GET THE INSTRUCTION
 / CLEAR THE LINK AND POSITION
 / SET THE BIT AND REPOSITION
 / PLACE IN THE MEMORY
 / GO GET THE NEXT VALUE
 / INCREMENT THE ADDRESS
 / GO BACK TO MICRO

/ THE FLASH ERROR ROUTINE
 / CLEAR THE ABSOLUTE ADDRESS
 / WAIT FOR A CLEAR KEYPAD
 / CLEAR THE DISPLAY
 / GET THE FIRST TIME CONSTANT
 / PLACE IN STORE
 / CLEAR THE TIMER
 / TIME OUT
 / JUMP -1
 / ISZ STORE
 / COUNT THE TIME CYCLES

INTERFIL

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 8-1

```
06744 5361 JMP -3
06745 1377 TAD TKB / GET THE SECOND TIME CONSTANT
06746 3152 DCA STORE / PLACE IN STORE
06747 4161 CALL / GET A SWITCH READING
06748 6269 SNOB / TEST FOR ANY KEYPRESS
06749 7440 SZA CLA / COUNT DOWN THE DISPLAY ON STORE
06750 5276 JMP TOZE
06751 2192 IRT STORE / GO AND FLASH AGAIN
06752 5367 JMP -5
06753 5352

06776 7775 TKA. 7775
06777 7720 TKB. 7720
```

/ CROSS THE PAGE BOUNDARY TO PAGE 8-2

*7000 *START+1000

```
07000 7333 AIOT. CLA CLL CML IAC RTR / SET THE AC TO 6000
07001 4161 CALL / PLACE IN THE MEMORY
07002 7941 PLACE / CLEAR THE ABSOLUTE ADDRESS
07003 3160 DCA SAVS / GET A HEX DIGIT FROM THE KEYPAD
07004 4161 CALL
07005 4441 HEX / PLACE IN TEMP
07006 3190 DCA TEMP / GET THE VALUE
07007 1150 TAD TEMP / MASK OUT BIT #8
07010 0211 AND CUS / TEST: IS THE VALUE > 7
07011 7450 COB. SNOB CLA / NO: GO TO THE ADDRESS LOAD
07012 5223 JMP SOB / YES: GET THE VALUE
07013 1150 TAD TEMP / ADD -11
07014 1220 TAD SNOB / TEST: IS IT A "C" KEYPRESS
07015 7450 SNOB CLA / YES: GO TO NEXT
07016 5471 SOT / NO: GO GET THE NEXT DIGIT
07017 5204 JMP AIOT+4

07020 7745 SNOT. 7745 / -11
07021 7000 SOC. 7000
07022 0777 CUS. 0777

07023 1400 SOB. TAD I SAVPC / GET THE INSTRUCTION
07024 0221 AND SOC / MASK OUT THE OP CODE
07025 4161 CALL / PLACE BACK IN THE MEMORY
07026 7941 PLACE
07027 1140 TAD SAVS / GET THE DEVICE CODE
07030 7104 CLL RAL / ROTATE IT OVER ONE OCTAL DIGIT
07031 7104 CLL RAL
07032 7104
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 8-2

```
07033 1150 TAD TEMP / ADD IN THE NEW DIGIT
07034 0222 AND CUS / BOUND THE CODE TO BITS #3 TO #11
07035 3160 DCA SAVS / PLACE THE NEW CODE IN SAVS
07036 1140 TAD SAVS / GET THE NEW CODE
07037 1400 TAD I SAVPC / COMBINE WITH THE INSTRUCTION
07040 4161 CALL / PLACE IN THE MEMORY
07041 7541 JMP AIOT+4
07042 5204
```

/ THE OPERATE GROUPS ASSEMBLY / ROUTINES

```
07043 1221 AOPR1. TAD SOC / SET THE AC EQUAL TO 7000
07044 4161 CALL / PLACE IN THE MEMORY
07045 7541 PLACE / GET A HEX VALUE FROM THE TABLE
07046 4161 CALL / GET A HEX READING FROM
07047 4441 HEX / THE KEYPAD
07050 1295 TAD QUM / ADJUST A POINTER TO THE TABLE
07051 3147 DCA POINT / PLACE IN POINT
07052 1547 TAD I POINT / GET THE POINTER FROM THE TABLE
07053 3147 DCA POINT / PLACE IN POINT
07054 5547 JMP I POINT / GO TO THE PROPER ROUTINE

07055 7056 QUM. QUM+1 / THE TABLE OF POINTERS
07056 7046 AOPR1+3
07057 7114 JAO
07060 7111 JAO
07061 7106 JAO
07062 7103 JAO
07063 7100 JAO
07064 7075 JAO
07065 7072 JAO
07066 7046 AOPR1+3
07067 7046 AOPR1+3
07070 7117 BSET11
07071 6745 SOT. NEXT
```

```
07072 4161 JA4. CALL / SET THE PROPER BITS IN THE
07073 7252 BSET4 / INSTRUCTION
07074 5244 JMP AOPR1+3

07075 4161 JA5. CALL
07076 7254 BSET5 / INSTRUCTION
07077 5244 JMP AOPR1+3

07100 4161 JA6. CALL
07101 7256 BSET6 / INSTRUCTION
07102 5244 JMP AOPR1+3
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 8-3

```
07103 4161 JA7. CALL
07104 7240 BSET7 / INSTRUCTION
07105 5244 JMP AOPR1+3

07106 4161 JA8. CALL
07107 7262 BSET8 / INSTRUCTION
07110 5244 JMP AOPR1+3

07111 4161 JA9. CALL
07112 7244 BSET9 / INSTRUCTION
07113 5244 JMP AOPR1+3

07114 4161 JA10. CALL
07115 7264 BSET10 / INSTRUCTION
07116 5244 JMP AOPR1+3

07117 1400 BSET11. TAD I SAVPC / GET THE INSTRUCTION
07120 7010 RAR / SET BIT #11
07121 7124 CLL CML RAL / PLACE IN THE MEMORY
07122 4161 CALL
07123 7541 PLACE / GO GET THE NEXT COMMAND
07124 5244 JMP AOPR1+3
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 9

```
07125 1237 AOPR2. TAD IOL2 / SET THE AC EQUAL TO 7400
07126 4161 CALL / PLACE IN THE MEMORY
07127 7541 PLACE / GET A HEX DIGIT FROM THE
07130 4161 CALL / KEYPAD
07131 4441 HEX / ADJUST TO POINT AT THE TABLE
07132 1365 TAD GUB / PLACE IN THE POINTER
07133 3147 DCA POINT / GET THE JUMP ADDRESS FROM THE TABLE
07134 1547 TAD I POINT / PLACE IN POINT
07135 3147 DCA POINT / GO TO THE PROPER ROUTINE
07136 5547 JMP I POINT

07137 7400 IOL2. 7400
```

/ SET THE PROPER BITS IN THE / INSTRUCTION

```
07140 4161 JB4. CALL
07141 7252 BSET4 / INSTRUCTION
07142 5330 JMP AOPR2+3

07143 4161 JB5. CALL
07144 7254 BSET5 / INSTRUCTION
07145 5330 JMP AOPR2+3

07146 4161 JB6. CALL
07147 7256 BSET6 / INSTRUCTION
07150 5330 JMP AOPR2+3
```

/ SET THE PROPER BITS IN THE / INSTRUCTION

```
07151 4161 JB7. CALL
07152 7260 BSET7 / INSTRUCTION
07153 5330 JMP AOPR2+3

07154 4161 JB8. CALL
07155 7262 BSET8 / INSTRUCTION
07156 5330 JMP AOPR2+3

07157 4161 JB9. CALL
07160 7264 BSET9 / INSTRUCTION
07161 5330 JMP AOPR2+3
```

/ SET THE PROPER BITS IN THE / INSTRUCTION

```
07162 4161 JB10. CALL
07163 7266 BSET10 / INSTRUCTION
07164 5330 JMP AOPR2+3

07165 7166 GUB. GUB-1 / THE TABLE OF POINTERS
07166 7151 JB7
07167 7157 JB9
07170 7130 AOPR2+1
07171 7130 AOPR2+3
07172 7130 AOPR2+3
07173 7112 B61
07174 7130 AOPR2+3
07175 7140 JB4
07176 7146 JB4
07177 7149 JB5
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 9-1

/ CROSS THE PAGE BOUNDARY TO PAGE 9-2

/ THE OPERATE GROUPS ASSEMBLY / ROUTINES

```
07200 7154 JBR
07201 6745 NEXT

07202 1214 AOPR3. TAD IOL3 / SET THE AC EQUAL TO 7401
07203 4161 CALL / PLACE IN MEMORY
07204 7541 PLACE / GET A HEX READING FROM
07205 4161 CALL / THE KEYPAD
07206 4441 HEX / ADJUST A POINTER TO THE TABLE
07207 1215 TAD BOR / PLACE IN POINT
07210 3147 DCA POINT / GET THE ADDRESS FROM THE TABLE
07211 1547 TAD I POINT / PLACE IN POINT
07212 3147 DCA POINT / GO TO THE PROPER ROUTINE
07213 5547 JMP I POINT
```

```
07214 7401 IOL3. 7401

07215 7216 BOB. BOB+1
07216 7205 AOPR3+3
07217 7205 AOPR3+3
07220 7205 AOPR3+3
07221 7205 AOPR3+3
07222 7205 AOPR3+3
07223 7205 AOPR3+3
07224 7205 AOPR3+3
07225 7232 JC4
07226 7240 JC7
07227 7235 JC5
07230 7205 AOPR3+3
07231 6745 NEXT
```

/ SET THE PROPER BIT IN THE / INSTRUCTION

```
07232 4161 JC4. CALL
07233 7252 BSET4 / INSTRUCTION
07234 5205 JMP AOPR3+3

07235 4161 JC5. CALL
07236 7254 BSET5 / INSTRUCTION
07237 5205 JMP AOPR3+3

07240 4161 JC7. CALL
07241 7260 BSET7 / INSTRUCTION
07242 5205 JMP AOPR3+3
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 9-2

/ THE BIT SET SUBROUTINES

```
07243 0002 AAA. 0002 / SET BIT #10
07244 0004 AAB. 0004 / SET BIT #9
07245 0010 AAC. 0010 / SET BIT #8
07246 0020 AAD. 0020 / SET BIT #7
07247 0040 AAE. 0040 / SET BIT #6
07250 0100 AAP. 0100 / SET BIT #5
07251 0200 AAG. 0200 / SET BIT #4
```

/ SET THE PROPER BIT IN THE / INSTRUCTION

```
07252 1251 BSET4. TAD AAG
07253 5267 JMP NBST

07254 1250 BSET5. TAD AAF
07255 5267 JMP NBST

07256 1247 BSET6. TAD AAE
07257 5267 JMP NBST

07260 1246 BSET7. TAD AAD
07261 5267 JMP NBST

07262 1245 BSET8. TAD AAC
07263 5267 JMP NBST

07264 1244 BSET9. TAD AAB
07265 5267 JMP NBST

07266 1243 BSET10. TAD AAA
```

```
07267 7521 NBST. SNOB / PLACE THE SET CONSTANT IN THE NO
07270 7300 CLA CLL / CLEAR THE AC AND LIMP
07271 1400 TAD I SAVPC / GET THE INSTRUCTION
07272 7501 RAR / OR IN THE BIT TO BE SET
07273 4161 CALL / PLACE IN THE MEMORY
07274 7541 PLACE / RETURN TO THE PROGRAM
07275 5544 RETURN
```

/ THE RINSAC ROUTINE / TO TOGGLE THE AC DISPLAY FLAG

```
07276 1133 RINSAC. TAD SWITCH
07277 7002 BSM / POSITION THE FLAG
07300 7004 RAL
07301 7030 CML RAR
07302 7002 BSM
07303 3133 DCA SWITCH / RESTORE THE SWITCH
07304 5544 RETURN / RETURN TO THE PROGRAM
```

/ REFER TO THE END OF THIS LISTING FOR / THE SOURCE OF THE DUMP PROGRAM / WHICH RESIDES IN THIS ADDRESS SPACE.

INTERCIL

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 10

/ CROSS THE PAGE BOUNDARY TO PAGE 97

07400 *START=1400

```

07400 7340 BIN.  CLA CLL CPA          / SET THE AC
07401 3133 DCA SWITCH        / SET THE SWITCH TO DISPLAY PC AND MD
07402 4161 CALL          / WAIT FOR A CLEAR KEYPAD
07403 4156 CLXP          /
07404 1227 TAD XJMP        / GET THE RETURN LINK INSTRUCTION
07405 3152 DCA STORE        / PLACE IT IN STORE
07406 1232 TAD PLUR          / GET THE LINK ADDRESS
07407 3153 DCA SHIFT        / PLACE IT IN SHIFT
07410 1400 TAD I SAVPC        / GET THE INSTRUCTION TO BE PERFORMED
07411 1234 TAD KAL          / ADD -CALL
07412 7450 BNA CLA          / TEST IS THE INSTRUCTION A CALL
07413 3554 JPP KALL          / YES: GO TO KALL
07414 1400 TAD I SAVPC        / NO: GET THE INSTRUCTION
07415 0230 AND PK          / MASK OUT THE OPCODE
07416 1231 TAD KIT          / ADD +4000
07417 7450 BNA CLA          / TEST: IS IT A JMS
07420 3546 JPP EJMS        / YES: EXECUTE A PSEUDO-JMS
07421 1400 TAD I SAVPC        / NO: GET THE INSTRUCTION
07422 0230 AND PK          / MASK OUT THE OPCODE
07423 1233 TAD KIT          / ADD -5000
07424 7450 BNA CLA          / TEST: IS THE INSTRUCTION A JPP
07425 3342 JPP EJP          / YES: EXECUTE A PSEUDO-JPP
07426 5263 JPP EXEC        / NO: GO TO THE NORMAL EXECUTE ROUTINE

07427 3553 KJMP.  JPP I SHIFT
07430 7000 PK.    7000
07431 4000 KIT.  4000
07432 7524 PLUR. RET
07433 3000 KAT.  3000
07434 3617 KAL.  -CALL
    
```

```

07435 1400 INAD. TAD I SAVPC        / GET THE INSTRUCTION
07436 0257 AND PK          / MASK OUT THE PAGE ADDRESS
07437 3144 DCA TIME          / PLACE IN TIME
07440 1400 TAD I SAVPC        / GET THE INSTRUCTION
07441 0261 AND PK          / MASK OUT THE CURRENT PAGE BIT
07442 7450 BNA CLA          / TEST: IS THIS CURRENT PAGE
07443 5250 JPP INDB        / NO: GO TO INDB
    
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 10-1

```

07444 1000 TAD SAVPC        / YES: GET THE CURRENT ADDRESS
07445 0260 AND PK          / MASK OUT THE PAGE NUMBER
07446 1144 TAD TIME          / COMBINE WITH THE PAGE ADDRESS
07447 3144 DCA TIME          / PLACE IN TIME

07450 1400 INDB. TAD I SAVPC        / GET THE INSTRUCTION
07451 0262 AND PK          / MASK OUT THE INDIRECT BIT
07452 7450 BNA CLA          / TEST: IS THIS AN INDIRECT
07453 3544 JPP RETURN        / NO: RETURN WITH THE ADDRESS
                                / IN TIME
07454 1544 TAD I TIME          / YES: GET THE TRUE ADDRESS
07455 3144 DCA TIME          / PLACE IN TIME
07456 3544 RETURN          / RETURN WITH THE ADDRESS IN TIME

07457 0177 HDI.    0177
07460 7400 PUD.  7400
07461 0200 OUT.  0200
07462 0400 LOT.  0400
    
```

```

07463 1000 EXEC. TAD SAVPC        / GET THE CURRENT ADDRESS
07464 7001 IAC          / GENERATE PC+1
07465 3144 DCA TIME          / PLACE IT IN TIME
    
```

```

                                / NOW WE CONTINUE WITH "NEXT"
                                / IN TIME

07466 1144 TAD TIME          / GET THE NEXT ADDRESS
07467 7001 IAC          / GENERATE NEXT+1
07470 3145 DCA SAVE          / STORE IN SAVE
07471 1544 TAD I TIME          / GET THE NEXT INSTRUCTION
07472 3154 DCA SAVI          / STORE IT IN SAVI
07473 1545 TAD I SAVE          / GET THE NEXT+1 INSTRUCTION
07474 3155 DCA SAV2          / PLACE IT IN SAV2
07475 1323 TAD TAL          / GET THE INSTRUCTION "JMS BACK"
07476 3544 DCA I TIME          / PLACE THE BREAKPOINT IN "NEXT"
07477 1323 TAD TAL          / GET "JMS BACK"
07478 3545 DCA I SAVE          / PLACE IT IN "NEXT+1"
07501 1323 TAD TAL          / GET THE BREAKPOINT
07502 7041 CIA          /
07503 1544 TAD I TIME          / ADD "NEXT" TO -BREAKPOINT
07504 3140 SZA CLA          / TEST: DID IT GET PLACED
07505 5722 JPP I NOT          / NO: MUST BE ROM OR SOMETHING
07506 1323 TAL          / GET THE BREAKPOINT
07507 1041 CIA          /
07510 1545 TAD I SAVE          / ADD "NEXT+1" TO -BREAKPOINT
07511 7440 SZA CLA          / TEST: DID IT GET PLACED
07512 5722 JPP I NOT          /
                                / IF WE GOT THIS FAR
                                / EVERYTHING MUST BE COOL SO
                                / WE WILL NOW EXECUTE THE INSTRUCTION
    
```

07513 1142 TAD SAVMD / RESTORE THE USER NO

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 10-2

```

07514 7421 HDL          /
07515 1141 TAD SAVFL        / RESTORE THE LINK
07516 7004 RAL          /
07517 7200 CLA          / CLEAR THE AC
07520 1140 TAD SAVAC        / RESTORE THE AC
    
```

```

07521 5400 JPP I SAVPC        / GO EXECUTE THE INSTRUCTION
    
```

```

07522 6434 HDI.  HALT          / BAD SIN
07523 4151 TAL.  JMS BACK        / BREAKPOINT INSTRUCTION

07524 3140 RET.  DCA SAVAC        / SAVE THE USER AC
07525 6004 OIF          / SAVE THE USER FLAGS
07526 3141 DCA SAVFL        /
07527 7521 BNP          / SAVE THE USER NO
07530 3142 DCA SAVMD        /
07531 1154 TAD SAVI          / GET THE ORIGINAL INSTRUCTION
07532 3544 DCA I TIME          / RESTORE IN "NEXT"
07533 1135 TAD SAV2          / GET THE ORIGINAL INSTRUCTION
07534 3545 DCA I SAVE          / RESTORE TO "NEXT+1"
07535 7340 CLA CLL CPA        / SET THE AC TO -1
07536 1151 TAD BACK          / GET THE RETURN ADDRESS
07537 3000 NIC.  DCA SAVPC        / RESTORE TO THE USER PC
07540 5741 JPP I .+1          /
07541 6400 SHELL          / GO TO THE MONITOR SHELL
    
```

```

07542 4161 EJPP. CALL          / GET THE NEXT ADDRESS INTO
07543 7435 INAD          / TIME
07544 1144 TAD TIME          / GET THE ADDRESS INTO THE AC
07545 5357 JPP HIC          / PLACE IT IN SAVPC AND END
                                / THE ROUTINE
    
```

```

07546 4161 EJMS. CALL          / GET THE NEXT ADDRESS
07547 7435 INAD          / INTO TIME
    
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 11

```

07550 1000 TAD SAVPC        / GET THE CURRENT ADDRESS
07551 7001 IAC          / INCREMENT IT
07552 3544 DCA I TIME        / PLACE IT IN THE NEXT LOCATION
07553 7001 IAC          / SET THE AC TO 0001
07554 1144 TAD TIME          / INCREMENT THE NEXT LOCATION
07555 5357 JPP HIC          / PLACE THE NEW ADDRESS IN SAVPC
    
```

```

07556 7305 KALL. CLA CLL IAC RAL / SET THE AC TO 0002
07557 1000 TAD SAVPC        / BUMP THE RETURN ADDRESS BY 2
07560 5263 JPP EXEC+2        / GO TO THE EXEC ROUTINE WITH
                                / THE ADDRESS OF THE RETURN
                                / POINT OF THE SUBROUTINE CALLED.
    
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 12

```

                                / THE PLACE ROUTINE
07561 3145 PLACE. DCA SAVE        / PLACE THE DATA IN SAVE
07562 1000 TAD SAVPC        / GET THE ADDRESS
07563 7430 BNA          / TEST: IS IT IS LOCATION 0000
07564 3544 RETURN          / YES: DO NOT LOAD AND RETURN
                                / WITH THE AC CLEAR
    
```

```

07565 1260 TAD PUD          / NO: ADD -177
07566 7700 BNA CLA          / TEST: IS THE PC > 177
07567 5374 JPP -.5          / YES: OK TO LOAD
07570 1000 TAD SAVPC        / NO: GET THE PC
07571 1377 TAD KHI43        / ADD -143
07572 7700 BNA CLA          / TEST: IS THE PC < 143
07573 5376 JPP .+3          / NO: DO NOT LOAD
07574 1145 TAD SAVE          / YES: OK TO LOAD
07575 3400 DCA I SAVPC        / LOAD TO THE USER MEMORY
07576 3564 RETURN          / RETURN TO THE PROGRAM

07577 7635 KHI43. 7635          / -143
    
```

/ THE FOLLOWING ROUTINES USE THE
/ PIE-WART INTERFACE

```

07578 6160 READI=6160
07579 6170 READ2=6170
07580 6161 WRITE1=6161
07581 6171 WRITE2=6171
07582 6162 SKIP1=6162
07583 6172 SKIP2=6172
07584 6163 SKIP3=6163
07585 6173 SKIP4=6173
07586 6164 RCRA=6164
07587 6165 MCRB=6165
07588 6175 MCRB=6175
07589 6174 MVR=6174
07590 6166 SPLAD1=6166
07591 6176 SPLAD2=6176
07592 6167 CPLAD1=6167
07593 6177 CPLAD2=6177
    
```

/ REENTER PAGE 02

06340 *START=340

```

06340 4007 IMPIE. CAP          / CLEAR ALL FLAGS
06341 4400 MCRB          / CLEAR THE DISPLAY
06342 1357 TAD KCRB        / GET THE CRA WORD
06343 6165 MCRB          / WRITE IT TO THE PIE
06344 7300 CLA CLL        /
06345 1360 TAD KCRB        / GET THE CRB WORD
06346 6175 MCRB          / WRITE IT TO THE PIE
06347 7300 CLA CLL        /
06350 1342 TAD KVR          / GET THE VECTOR HANDLER POINTER
06351 6174 MVR          / LOAD IT TO THE VECTOR REGISTER
06352 7300 CLA CLL        / CLEAR THE AC AND LINK
06353 1361 TAD KTTY        / GET THE WART CONTROL WORD
06354 6171 WRITE2          / WRITE TO THE WART
06355 7300 CLA CLL        / CLEAR THE AC AND LINK
06356 5564 RETURN          / GO BACK TO THE PROGRAM
    
```

/ THE PIE INITIALIZE ROUTINE

```

06357 7200 KCRB. 7200
06360 1560 KCRB. 1560
06361 7600 KTTY. 7600
06362 0200 KVR. 0200
    
```

/ CROSS TO PAGE 06. THE LAST PAGE

*7400 *START=1400

```

                                / THE TTY OUTPUT ROUTINE
07600 6163 TALK. SKIP2          / SKIP ON CLEAR INIT BUFFER
07601 3200 JPP I -1          / INIT BUFFER NOT CLEAR YET
07602 6161 WRITE1          / WRITE THE AC TO THE WART
                                / INIT BUFFER
07603 3144 DCA TIME          / CLEAR THE AC AND STORE THE
                                / OLD CHARACTER IN TIME
07604 3564 RETURN          / RETURN TO THE PROGRAM
    
```

/ THE LISH ROUTINE TO GET A CHARACTER

INTERFIL

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 12-2

/ FROM THE TTY KEYBOARD

```
07605 6172 LISH. SKIP3 / RESET THE START BIT SENSE FLAG
07606 7003 NOP / SET THE READER RUN FLAG
07607 6166 SFLAG1 / WAIT FOR THE FIRST START BIT
07610 6172 SKIP3 / NOT YET
07611 5210 JMP -1 / CLEAR THE READER RUN FLAG
07612 6167 /
07613 6162 READ. SKIF1 / WAIT FOR DATA READY FLAG
07614 5213 JMP -1 / NOT YET
07615 7200 CLA / CLEAR THE AC
07616 6160 READ1 / READ THE LUNTY BUFFER INTO
/ THE AC
07617 0221 AND TTYM / MASK OUT THE UNWANTED BITS
07620 5564 RETURN / GO BACK TO THE PROGRAM
07621 0377 TTYM. 0377 / TTY MASP
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 13

/ THE BIN LOADER FOR MONITOR

```
/ THIS PROGRAM LOADS TAPES IN BIN
/ FORMAT GENERATED BY THIS MONITOR
/ OR BY INTERFIL OR DEC ASSEMBLERS.
/ (SUCH AS IFDOS PAL, FOPAL III, PAL9,
/ PAL III, ETC.) THIS LOADER IGNORES
/ ALL CHANGE FIELD INSTRUCTIONS ON
/ THOSE TAPES. IT ALSO ECHOS
/ ALL CHARACTERS BETWEEN RUBOUTS ON
/ THOSE TAPES.
```

/ THESE SYMBOLS ARE FOR THE BIN LOADER

```
0156 LT=SAV3
0160 CHRSLUMSAVS
0157 LAST=SAV4
0150 FIRST=TEMP
0151 SEC=BACK
0152 THIRD=STORE
0153 DATA2=SHIFT
0154 PC2=SAV1
```

/ THE LOADER ALSO USES HOLD AND SAVPC

```
07622 4161 BIN. CALL / RESET THE PIE-ART CARD
07623 6340 IMPIE / AND CLEAR CHRSLUM
07624 5160 DCA CHRSLUM
07625 3151 DLA SEC / CLEAR SEC
07626 7040 CMA / SET LT TO 7777
07627 3156 DCA LT / SET THE AC TO 200
07630 1371 TAD K200 / SET LAST TO AN LT CHAR
07631 3157 DCA LAST / SET THE AC TO 102
07632 1374 TAD K102 / MAKE THE FIRST CHAR AN ORIGIN
07633 3150 DCA FIRST / EQUAL TO 0200
07634 7340 BEGG. CLA CLL CMA / SET THE AC
07635 3153 DCA DATA2 / SET DATA2
07636 3154 BEG. DCA PC2 / CLEAR PC2
07637 4161 CALL / GET THE FIRST CHARACTER FROM TTY
07640 7605 LISH / STORE IN HOLD
07641 3146 DCA HOLD / GET THE FIRST CHAR
07642 1146 TAD HOLD / TEST: IS IT A RUBOUT?
07643 1373 TAD #RUB / YES: GO TO A RUBOUT ROUTINE
07644 7706 JMP RUB / NO: GET THE CHAR
07645 5306 JMP MORE / TEST: IS IT AN LT CHAR
07646 1146 TAD HOLD / GET THE CHAR
07647 1367 TAD #CMB / TEST: IS IT AN LT CHAR
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 13-1

```
07650 7630 SNA CLA / YES: GO TO THE LT ROUTINE
07651 5343 JMP LTC / NO: STORE THE CHAR IN LAST
07652 1146 TAD HOLD / TEST: HAVE WE HAD AN LT YET?
07653 3157 DCA LAST / NO: IGNORE THE CHAR
07654 1156 TAD LT / ARE WE IN THE MIDDLE OF A PC
07655 7640 SZA CLA / LOAD?
07656 5236 JMP MORE / NO: GO ON TO MORE
07657 1146 TAD HOLD / YES: GET THE ADDRESS
07660 1372 TAD #D / COMBINE WITH THE SECOND HALF
07661 7700 SNA CLA / STORE THE NEW ADDRESS
07662 5236 JMP BEG / GO TO CHAR UPDATE
07663 1146 TAD HOLD / BETWEEN RUBOUTS WE ECHO THE
07664 1160 TAD CHRSLUM / CHARS READ
07665 3160 DCA CHRSLUM
07666 1150 TAD FIRST / GET THE FIRST CHAR FROM THE
/ PREVIOUS PAIR OF CHARS
07667 0370 AND KLONG / TEST: IS IT AN ORIGIN?
07670 7640 SZA CLA / YES: DO TO PC LOAD
07671 5352 JMP PCL / ARE WE IN THE MIDDLE OF A PC
/ LOAD?
07672 2154 ISZ PC2 / NO: GO ON TO MORE
07673 3311 JMP MORE / YES: GET THE ADDRESS
07674 1000 PCL2. TAD SAVPC / COMBINE WITH THE SECOND HALF
07675 1151 TAD SEC / STORE THE NEW ADDRESS
07676 3050 DCA SAVPC / GO TO CHAR UPDATE
07677 5204 JMP BEG
07700 4161 RUB. CALL / BETWEEN RUBOUTS WE ECHO THE
07701 7605 LISH / CHARS READ
07702 4161 CALL / GET THE CHAR FROM TIME
07703 7600 TALK / TEST: IS IT A RUBOUT?
07704 1144 TAD TIME / IF A RUBOUT IT WILL SET AC TO ZERO
07705 1373 TAD #RUB / YES: GO ON WITH THE LOADING
07706 7700 SNA CLA / NO: CONTINUE TO IGNORE CHARS
07707 5237 JMP RUB
07710 5300 JMP MORE
07711 2153 MORE. ISZ DATA2 / ARE WE IN THE MIDDLE OF
/ A DATA LOAD SEQUENCE?
07712 5316 JMP DL2 / YES: GO LOAD THE SECOND PART
07713 1146 TAD HOLD / NO: GET THE CHAR
07714 3152 DCA THIRD / LOAD TO THIRD
07715 5236 JMP BEG / GO GET ANOTHER CHAR
07716 1150 DL2. TAD FIRST / GET THE FIRST CHAR
07717 7002 BSM / POSITION
07720 1151 TAD SEC / GET THE SECOND HALF
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 13-2

```
07721 4161 CALL / PLACE THE COMPLETE WORD
07722 7561 PLACE / IN THE MEMORY
07723 2000 ISZ SAVPC / INCREMENT THE ADDRESS
07724 7300 BMO. CLA CLL / CLEAR THE AC AND LUNTY
07725 1152 TAD THIRD / MOVE UP ONE PAIR OF CHARS
07726 3150 DCA FIRST / GO GET ANOTHER CHAR
07727 1146 TAD HOLD / LOAD THE ORIGIN TO THE ADDRESS
07730 3151 DCA SEC / MASK OUT THE CHANNEL 7 PUNCH
07731 5234 JMP BEG / LOAD THE FIRST HALF TO SAVPC
07732 1150 PCL. TAD FIRST / CLEAR FIRST
07733 0366 AND PMSK / UPDATE THE CHAR
07734 7002 BSM / SET PC2
07735 3000 DCA SAVPC / CLEAR LT
07736 3150 DCA FIRST / GET THE LAST CHAR RECEIVED
07737 1146 TAD HOLD / TEST: WAS THE LAST CHAR AN LT
07740 3152 DCA THIRD / YES: GO GET ANOTHER CHAR
07741 7040 CMA / NO: END OF THE LOAD
07742 5236 JMP BEG / COMPUTE THE CHMSUM, MASK OUT
/ STRAY BIT WHICH APPEARS ON SOME
/ PAL9-9 GENERATED TAPES.
07743 3156 LTC. DCA LT / ADD TO THE ACCUMULATED SUM
07744 1157 TAD LAST / NOW COMPENSATE FOR ADDIN-
07745 1367 TAD #CMB / THE LAST TWO CHARS TWICE
07746 7650 SNA CLA / STORE THE RESULT IN THE AC
07747 5234 JMP BEG / SET THE USER PC TO 140
07750 1150 TAD FIRST / GO TO HALT
07751 0366 AND PMSK / THE TABLE OF CONSTANTS
07752 7002 BSM / PROGRAM. THE PROGRAM IS TO BE USED
07753 1151 TAD SEC / IN THE USER RUN MODE BY THE USER
07754 7041 CJA / TO DUMP SECTIONS OF MEMORY OUT ONTO
/ PAPER TAPE IN THE BINARY 'BIN'
07755 1160 TAD CHMSUM / FORMAT FOR OFFLINE STORAGE.
07756 7041 CJA / THESE TAPES MAY LATER BE LOADED USING
07757 1150 TAD FIRST / THE BIN LOADER IN THE MONITOR PROGRAM
07760 1151 TAD SEC / BY PRESSING CONTROL FOLLOWED BY PRESSING
07761 3140 DCA SAVPC / OUR OPERATOR SYSTEM RESTART.
07762 1375 TAD #140 / TO SET THE ADDRESS RANGE OF THE SECTION
07763 3000 DCA SAVPC / YOU WANT PUNCHED. PLACE THE ADDRESS
07764 5765 MALT / OF THE FIRST WORD OF THE BLOCK IN
/ LOCATION 131. PLACE THE ADDRESS OF
/ LAST WORD OF THE BLOCK IN LOCATION 132.
/ THE DUMP PROGRAM WILL PUNCH OUT A BIN
/ TAPE WITH LEADER/TRAILER. ALL LOCATIONS
/ FROM FIRST TO LAST AND A CHECKSUM. THE
/ TAPE WILL END WITH ANOTHER SECTION OF
/ LEADER/TRAILER.
07766 0077 PMSK. 0077 / SET THE AC TO 7777
07767 7600 #CMB. 7600 / DISABLE THE CF TYPED
07770 0100 #DUM. 0100 / INITIALIZE THE PIE-ART CARD
07771 0200 #200. 0200 / SET THE AC EQUAL TO 7776
07772 7500 #FD. 7500 / PLACE THIS IN BACK
07773 7401 #RUB. 7401 / CLEAR THE CHECKSUM
07774 0102 #102. 0102 / PUNCH OUT LEADER/TRAILER
07775 0140 #140. 0140 / PUNCH THE ORIGIN
/ POSITION FOR THE FIRST BITS
/ MASK OUT THE BITS
/ ADD THE CHANNEL 7 PUNCH
/ PUNCH IT ON THE TTY
/ GET THE SECOND HALF OF THE ORIGIN
/ PUNCH IT
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 14

/ REENTER PAGE #0

*7305 *START+1305

```
/ THIS IS THE MEMORY DUMP USER
/ PROGRAM. THE PROGRAM IS TO BE USED
/ IN THE USER RUN MODE BY THE USER
/ TO DUMP SECTIONS OF MEMORY OUT ONTO
/ PAPER TAPE IN THE BINARY 'BIN'
/ FORMAT FOR OFFLINE STORAGE.
/ THESE TAPES MAY LATER BE LOADED USING
/ THE BIN LOADER IN THE MONITOR PROGRAM
/ BY PRESSING CONTROL FOLLOWED BY PRESSING
/ OUR OPERATOR SYSTEM RESTART.
/ TO SET THE ADDRESS RANGE OF THE SECTION
/ YOU WANT PUNCHED. PLACE THE ADDRESS
/ OF THE FIRST WORD OF THE BLOCK IN
/ LOCATION 131. PLACE THE ADDRESS OF
/ LAST WORD OF THE BLOCK IN LOCATION 132.
/ THE DUMP PROGRAM WILL PUNCH OUT A BIN
/ TAPE WITH LEADER/TRAILER. ALL LOCATIONS
/ FROM FIRST TO LAST AND A CHECKSUM. THE
/ TAPE WILL END WITH ANOTHER SECTION OF
/ LEADER/TRAILER.
```

```
07305 7340 DUMP. CLA CLL CMA / SET THE AC TO 7777
07306 6402 / DISABLE THE CF TYPED
07307 4161 CALL / INITIALIZE THE PIE-ART CARD
07310 0340 IMPIE / SET THE AC EQUAL TO 7776
07311 7344 CLA CLL CMA RAL / PLACE THIS IN BACK
07312 3151 DCA BACK / CLEAR THE CHECKSUM
07313 3160 DCA SAV5 / PUNCH OUT LEADER/TRAILER
07314 4161 CALL / PUNCH THE ORIGIN
07315 6363 TWTY / POSITION FOR THE FIRST BITS
07316 1131 TAD HOLD2 / MASK OUT THE BITS
07317 7002 BSM / ADD THE CHANNEL 7 PUNCH
07320 0367 AND HALF / PUNCH IT ON THE TTY
07321 1370 TAD ORIGIN / GET THE SECOND HALF OF THE ORIGIN
07322 4161 CALL / PUNCH IT
07323 7371 PUNCH / MASK OUT THE BITS
07324 1131 TAD HOLD2 / ADD THE CHANNEL 7 PUNCH
07325 0367 AND HALF / PUNCH IT ON THE TTY
07326 4161 CALL / PUNCH IT
```

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 14-1

```
07327 7371 PUNCH / GET THE DATA WORD
07330 1531 DATAL. TAD 1 HOLD2 / PUNCH THE FIRST HALF OF THE
07331 2151 ISZ BACK. / SECOND HALF
07332 7002 BSM / FIRST: POSITION THE BITS
07333 0367 AND HALF / MASK OUT THE BITS
07334 4161 CALL / PUNCH ON THE TTY
07335 7371 PUNCH / GET THE FIRST FLAG
07336 1151 SZA CLA / TEST: IS THIS THE FIRST
07337 7640 JMP DATAL. / YES: GO BACK FOR THE SECOND HALF
07341 7344 CLA CLL CMA RAL / NO: RESET BACK TO 7776
07342 3151 DCA BACK / GET THE ADDRESS POINTER
07343 1131 TAD HOLD2 / NEGATE IT
07344 7041 CJA / ADD THE END ADDRESS
07345 1132 TAD HOLD2 / INCREMENT THE ADDRESS POINTER
07346 2131 NOP / IN CASE HOLE 12 IS EQUAL TO 7777
07347 7000 SZA CLA / TEST: ARE THEY THE SAME?
07350 6402 JMP DATAL. / NO: NOT DONE YET. GO ON
07351 5330 / YES: PUNCH OUT THE CHMSUM
07352 1160 CHSUM. TAD SAV5 / FIRST OF SECOND HALF
07353 2151 ISZ BACK. / FIRST HALF
07354 7002 BSM / MASK OUT THE BITS
07355 0367 AND HALF / PUNCH ON THE TTY
07356 4161 CALL / GET THE FIRST FLAG
07357 7600 TALP / DONT YET
07360 1151 TAD BACK / GET THE ADDRESS POINTER
07361 7640 SZA CLA / DONT YET
07362 3352 JMP CHSUM / NOT YET
07363 4161 CALL / ALL DONE. PUNCH OUT LEADER/TRAILER
07364 6363 TWTY / RESTORE THE CF REQUEST TIMES
07365 6402 AND / END OF PROGRAM
07366 7402 MLT /
07367 0077 HALF. 0077 / THE CONSTANTS
07370 0100 ORIGIN. 0100 /
07371 4161 PUNCH. CALL / OUTPUT THE 80 TO THE TTY
07372 7600 TALP / GET THE UNCHARACTERS
07373 1144 TAD TIME / COMBINE WITH THE CHECKSUM
07374 1160 TAD SAV5 / GO BACK TO THE PROGRAM
07375 3160 DCA SAV5
07376 5564 RETURN
```

/ REENTER PAGE #:

INTERFIL

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 14-2

46363 *START+363

06362	1373	TMTV.	TAD	1963	/ SET THE AC TO MINUS 63
06364	3165		DOA	SAVE	/ PLACE TM SAVE
06365	1374		TAD	PLT	/ GET AN LT CHAR
06366	4161		CALL		/ PUNCH IT ON THE TTY
06367	7460		TALK		
06370	2145		ISZ	SAVE	/ DONE PUNCHING YET
06371	5365		JMP	-4	/ NOT YET
06372	8544		RETURN		/ ALL DONE. RETURN TO PROGRAM
06373	7700	KN63.	7700		
06374	0200	PLT.	0200		

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 15

AAA	7243	DISP1	0134	KCALLY	6043	PRSH	7766	SWITCH	0132
AAE	7244	DISP2	0135	KCAR	7767	POINT	0147	TAE	6517
AAE	7245	DISP3	0136	KORA	6257	POW	4524	TADJ	6146
AAE	7246	DISP4	0137	KCRB	6360	PUD	7460	TAL	7523
AAE	7247	DL2	7716	KFD	7772	PUMP	6723	TALJ	7600
AAE	7250	DUWP	7305	KIT	7421	PUNCH	7371	TCNT	4224
AAE	7251	EJPP	7542	KJPP	7427	RAM2J	6050	TEMP	0150
AAE	4443	EJRS	7546	KLINC	7670	RAM2Y	6047	THIRD	0152
AC	0146	ELO	6663	KLONG	7770	RAM2Z	6046	TMRU	6111
ACDIS	6154	EXEC	7463	PLT	6374	RRA	6164	TIME	0144
ADCA	6433	EXIT	6051	KH43	7577	REAC	7615	TRA	6776
ADJT	4367	FAD	6177	KH63	6373	READ1	6160	TRE	6777
ADTS	4706	FIRST	0150	KH7	6473	READ2	6170	TI	6223
ATNC	6424	FLSH	6752	KMB	6473	REFSH	6105	TOZ	6670
ATOT	7000	GOON	6560	KRETY	6044	RESET	6145	TOZE	6676
ATSI	6431	OOT0	6407	KRUB	7773	RET	7524	TYM	7621
AJPP	4440	QUB	7165	KTTY	6361	RETURN	8564	TUC	6790
AJMS	6436	QUM	7095	KVR	6362	RETS	0144	TUGJ	6751
ADPR1	7043	OUT	7461	K1000	6713	RETY	6075	UDIS	6134
ADPR2	7125	HALF	7367	K102	7774	RINSAC	7276	UG	6424
ADPR3	7202	HALT	6434	K140	7775	RJW	7700	UJ	6424
ATAD	6427	HEX	6441	K200	7771	RUN	6436	UAVE	6570
BACK	0151	HEUC	6444	K3000	6635	SAVAC	0140	URRA	6165
BAQ	7724	HIC	7537	K3000	6442	SAVE	0145	URB	6175
BASE	6045	HOLD	0146	LAST	0157	SAVFL	0141	WRITE1	6161
BEG	7436	HOLD1	0130	LISN	7605	SAVMO	0142	WRITE2	6171
BEGG	7434	HOLD2	0131	LOT	7462	SAVPC	9090	WFR	6174
BJN	7422	HOLD3	0132	LT	0156	SAV1	0154	YED	6667
BLF	6474	HOT	7522	LTC	7743	SAV2	0155	ZOL1	7137
BOB	7215	IMD	7470	MBS	6950	SAV3	0156	ZOL2	7214
BSET10	7264	INDB	7450	MBSY	7267	SAV4	0157	ZON	6735
BSET11	7117	INIT	6050	MDD15	6151	SAV5	0158	ZOT	6453
BSET4	7252	INPIE	6340	RICRO	6490	SET	0161		
BSET5	7254	INSAC	6425	NR	7430	SETFC	6543		
BSET6	7256	JAI0	7114	NOKE	7711	SPLAC1	6166		
BSET7	7260	JAA	7072	RRRA	6444	SPLAC2	6171		
BSET8	7262	JAS	7075	MS1	6316	SNELL	6400		
BSET9	7264	JAK	7106	MSC2	6317	SHIFT	0153		
BUG	6423	JAT	7105	MSC3	6320	SHIFTY	2322		
CALL	4161	JAB	7106	MSC4	6321	SIN	7400		
CALLY	0161	JAP	7111	MS1	6149	SIP1	6162		
CALLY	6044	JB10	7162	MS2	6150	SIFL	6161		
CFLAG1	6167	JB4	7140	NET	6745	SIF3	6172		
CFLAG2	6177	JB5	7142	NOT	7457	SIP4	6171		
CHKSUM	0140	JBZ	7146	ODSLO	6305	SNERD	4507		
CHSUM	7392	JB7	7151	OW	6171	SNPT	7020		
CLYPO	6156	JB8	7154	PA3	6463	SOE	7022		
COE	7011	JB9	7157	SPIN	6790	SOE	7021		
CRUMP	6176	K4	7232	WPT	6455	SOY	7021		
CYS	7022	K5	7236	PIC	6540	STAL	0148		
DATAL	7397	K7	7240	PCL	7021	STAGT	0150		
DAT2	6177	K8	7242	PCL2	6734	STATUS	0147		
D1	674	KAL	744	PCL3	0154	STIME	0149		
DEFF	7470	KAL1	7470	PLAGE	7471	SWAP	7471		
DEFF	7470	KAT	7471	PLWH	7472	SWDE	7472		

/ MONITOR 2 IFDOS PAL 1A 06-APR-77 PAGE 16

NO ERRORS DETECTED
 NO LINKS GENERATED
 252 SYMBOLS
 6K MEMORY UTILIZED

CHAPTER 9
INTERCEPT JR. AUDIO CARD

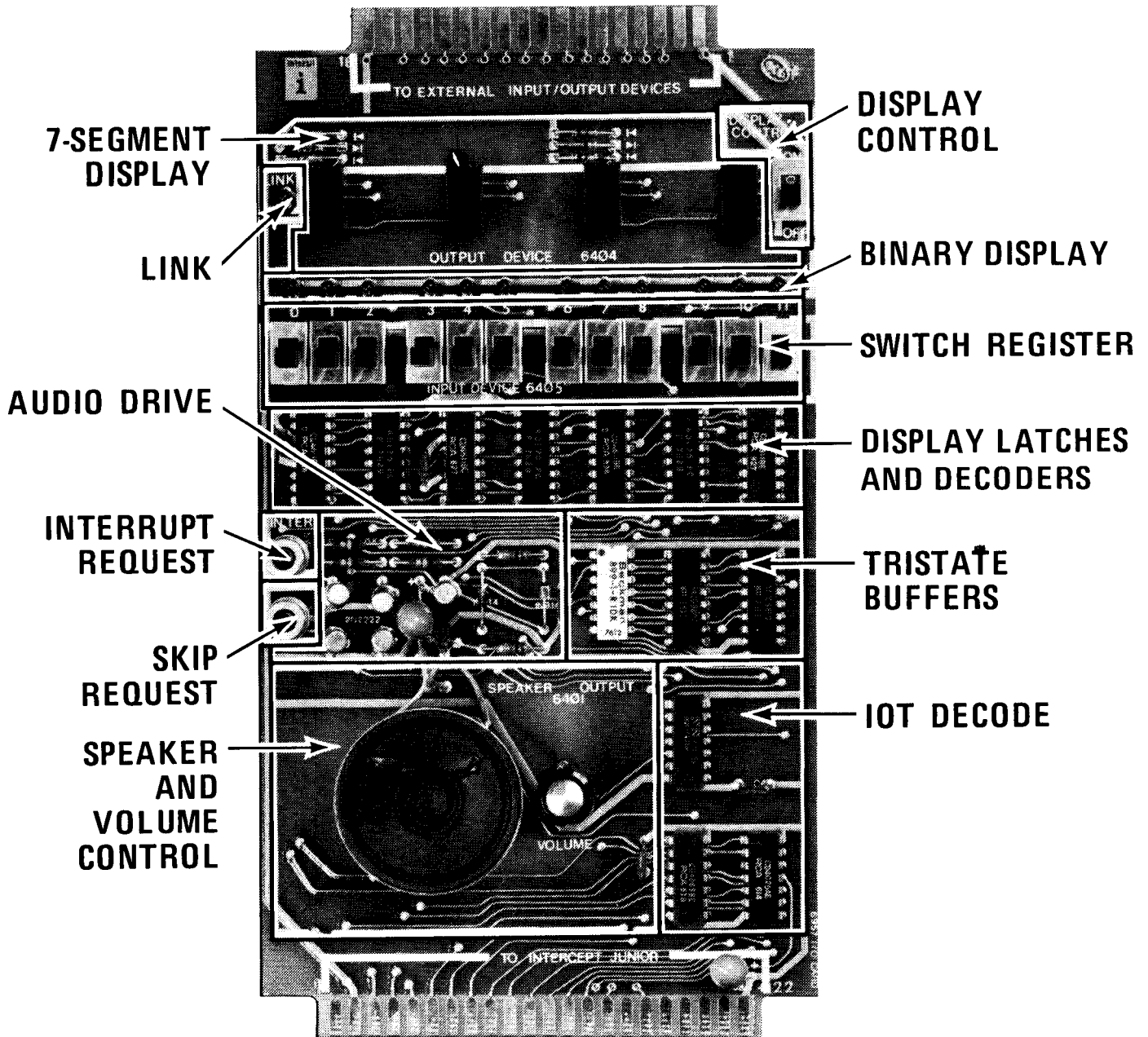


FIGURE 9-1

INTRODUCTION

The INTERCEPT JR. AUDIO MODULE, 6957-AUD/VIS, pictured in Figure 9-1, is used in microprocessor tutorial courses developed by INTERSIL INC.

The user can "click" the speaker or produce tones by controlling the rate at which the speaker clicks; the user can read a switch register and load data to an LED display register in either binary or in both binary and octal.

DISCUSSION

The AUDIO card makes use of the three unused IOT instruction codes 64X1, 64X4 and 64X5 brought out to connector pins Y, C and 15 of the INTERCEPT JR. module.

The card should be plugged in with the LED display on top and the speaker below using the card edge connector designated "to INTERCEPT JUNIOR".

The switch register is connected to the DX bus via two 340098 three-state hex buffers. The LED binary register is driven by three 74C175 quad D-latches with their inputs connected to the DX bus. The true outputs of the latches drive three 4511 BCD to 7 segment decoder drivers. The D input of each of the 4511's is grounded so that the seven segment display can only display in octal. The display can be blanked by pulling the blanking inputs on the 4511's low via the Display Control Switch S₁₂.

All the switch outputs are pulled up to V_{CC} via the 10K resistor pack.

IOT 6401 along with $\overline{\text{DEVSEL}}$ and XTC drives a 4025 three input NOR so that during $\text{IOTA} \cdot \overline{\text{DEVSEL}} \cdot \text{XTC}$ the 74C74 flip-flop is clocked by the execution of this instruction. The flip-flop toggles every time it is clocked as its $\overline{\text{Q}}$ output is connected back to the D input. This turns the transistors in the push-pull driver alternately ON or OFF, charging and discharging the 68 microfarad capacitor through the speaker voice coil and producing an audible click.

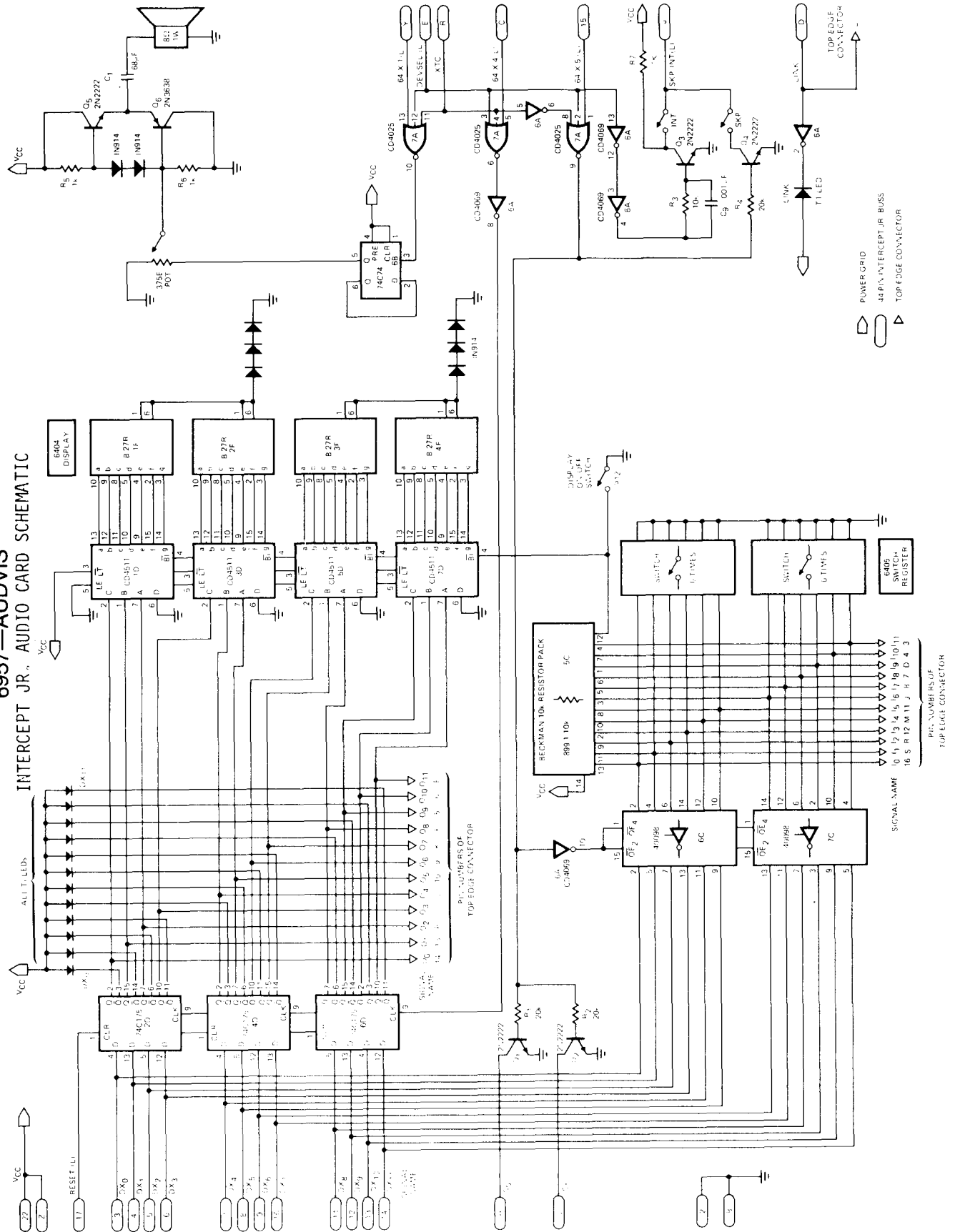
IOT 6404 is also an output instruction and thus is gated with $\overline{\text{DEVSEL}}$ and XTC to produce a load pulse (inverted by a 4069) to the three quad D-latches connected to the DX bus. The latches will thus store the contents of the AC which are placed on the bus by the IM6100 during $\text{IOTA} \cdot \overline{\text{DEVSEL}} \cdot \text{XTC}$.

IOT 6405 is an input instruction and is decoded along with $\overline{\text{DEVSEL}}$ and XTC to produce a strobe pulse at $\text{IOTA} \cdot \overline{\text{DEVSEL}} \cdot \text{XTC}$ time. This pulse is inverted by a 4069 and enables the tristate buffers onto the DX bus and also turns ON the two 2N2222 transistors driving the C₀ and C₁ lines. The IM6100 thus reads the DX bus during $\text{IOTA} \cdot \overline{\text{DEVSEL}} \cdot \text{XTC}$ and loads the data into the accumulator.

The INTREQ and SKP lines to the IM6100 are multiplexed onto the same line. The data read strobe generated by an IOT 6405 enables the SKP line so that depression of the SKP switch will drive the SKP line low. The INTREQ line is always enabled except during $\overline{\text{DEVSEL}}$ time. Actually, the SKP line is sampled only during $\overline{\text{DEVSEL}} \cdot \text{XTC}$, but for simplicity, interrupt requests are disabled even during $\overline{\text{DEVSEL}} \cdot \text{XTC}$. In any case, the INTREQ line is sampled only during the last cycle of an instruction execution during the first major state time.

The LINK bit drives an LED diode directly via a 4069.

6957—AUDVIS INTERCEPT JR., AUDIO CARD SCHEMATIC



CHAPTER 10

INTERCEPT JR CASSETTE INTERFACE CARD

INTRODUCTION

The INTERCEPT JR AUDIO CASSETTE INTERFACE MODULE, 6954-AC1 is pictured in Figure 10-1.

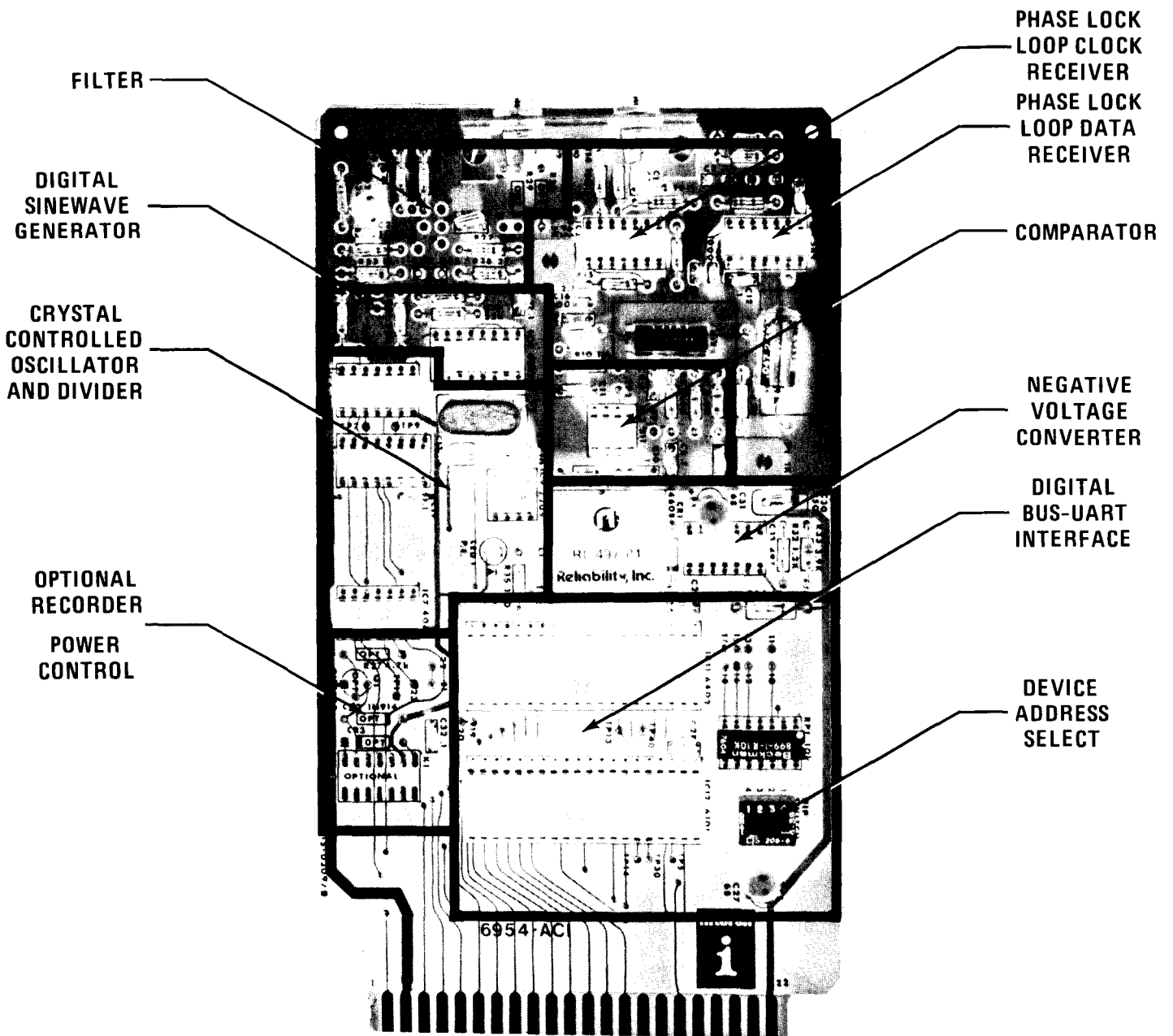


Figure 10-1

The 6954 Cassette Interface module allows the Intercept Jr. user to store programs on an inexpensive cassette tape recorder. The recording technique used is a variation of the proposed "Byte Magazine" standard. In addition to this standard signal, a multiple of the data clock is recorded on the tape. The data and the clock are recovered from the tape by using phaselock loops. The use of phaselock loops makes the system insensitive to amplitude variations, noise, and A.C. line interference. The self-clocking feature allows the system to operate independent of tape speed variations. Data is recorded and played back at 300 baud rate. Thus, approximately 200,000 characters may be recorded on a standard two hour cassette.

The record/playback system is shown in Figure 10-2.

The system can be subdivided into three main sections:

1. Transmitter section consisting of a clock, divider, clock gating, digital frequency generator - mixer and low pass filter.
2. Receiver section consisting of two phaselock loops and a comparator.
3. DC-DC converter to generate -5v from the +5v Jr. supply.
4. Digital section composed of a UART and a PIE chip.

The transmitter section takes the serial data being transmitted from the UART and converts it to standard frequency shift signals -- a higher frequency (Mark) for a digital "1" and a lower frequency (Space) for a digital "0". The ones and zeroes at the UART transmit line are converted to a series of marks and spaces on the magnetic tape. A tape cassette designed specifically for a digital system would write ones and zeroes directly on the tape by using a tape saturation techniques. The record system on an audio cassette

is not designed to operate in the saturation mode. Therefore, digital information is best stored in the form of two different frequencies, both within the frequency range of the recording system. This recording system is similar to techniques used to transmit data over telephone lines. Where telephone lines have phase distortion (harmonics arriving out of phase with the fundamental frequency), tape systems have speed variation problems. The speed variation problem in this system is resolved by recording the clock with data on the tape.

The receiver section has two phaselock loops. One loop is locked to the data mark and space frequencies. The DC control voltage for the Voltage Controlled Oscillator (VCO) is sensed by a comparator circuit. As the input frequency varies, so does the DC control voltage, as a result, the comparator output is a one for the mark frequency and a zero for a space frequency. The ones and zeroes are sent to the UART as received data.

Another PLL (phaselock loop) locks onto the clock signal and supplies the UART with a receiver clock. Any speed variations in the tape recorder are tracked by the data and clock PLLs, thereby making the system immune to speed variation.

The digital section consists of a standard Intersil UART (Universal Asynchronous Receiver Transmitter) and PIE (Parallel Interface Element).

THE RECORDER

Practically any cassette recorder can be used in this system. A number of different brands of recorders have been successfully used to record and play-back digital data. There may exist certain recorders whose circuitry is so poor as to be unusable in this system. Some recorders may have excessive AC hum which will create data errors. Sometimes the hum problem can be reduced by reversing the line plug. If a recorder with hum problems has provisions for batteries, you can run the unit from batteries to eliminate the hum problem.

The MIC output from the 6954-ACI board is a low-level signal which should be connected to the microphone input jack of the recorder. Most recorders label the microphone input "MIC". A shielded cable should always be used for the MIC interconnection. Failure to use a shielded cable may cause noise to be recorded along with the useful signal. Numerous jumper pads are provided near the 6954 MIC jack to allow the user to tailor the output to any special application. The 'from the factory' jumper set-up sets the output level to an optimum level for most recorders. If you have a recorder with an 'AUX' input, which is an input meant for a high-level signal, you may want to use the AUX input instead of the MIC input. The use of the AUX input would only be necessary in an extremely noisy environment. To create a high-level output from the 6954 board, cut the trace between jumper pads 2 and 3 and then connect pad 2 to pad 1. An even higher signal can be obtained by connecting pad 2 to TP10.

Some recorders have a monitor feature in which the signal being recorded is amplified and sent out on the earphone (EAR) jack during the recording process. This may be a quality signal which can be used to test and set-up the receiver portion of the board. Usually there is no volume control for this monitor signal and in some cases it may not be of sufficient level or quality to be used as a valid receiver signal. If a good monitor signal is not available, all testing should be done in two passes -- the first to record a signal and the second to play the signal back to the receiver section.

During data playback, the setting of the volume control is not critical. Unplug the EAR plug on the recorder and adjust the volume to a normal listening level -- this position of the control can then be marked for future reference. If at any time the system will work only for a narrow range of volume settings, a system problem exists and should be debugged. If a scope is available, the volume control should be adjusted so that a one volt peak-peak signal is present at the EAR jack of the 6954 board.

Although inexpensive cassette cartridges are usable since tape quality is not a significant factor, the mechanical quality of the better cassettes help reduce jamming problems that cassette recorders experience.

A recorder that has a "tape counter" feature is very useful for locating the approximate position of data stored on a tape.

SOURCES OF NOISE INTERFERENCE

The possibility of either recording errors or playback errors exists when there is an excessive amount of noise interference present. Possible sources of noise in this system are ground loops, AC line interference, RF interference, and supply voltage transients.

Ground loops are currents flowing through the ground traces, shields, and recorder frame which may create false signals either on the board or in the recorder. If an Intercept system has external equipment connected to it, improper grounding techniques can create system problems. A good method for dealing with grounding problems is to first draw a block diagram of all components in the system and then draw all ground connections that are present. An analysis can then be made of the path that ground currents must take.

AC line interference can be caused either by equipment connected to the system or by proximity to sources of AC voltages. An AC problem caused by a piece

of peripheral equipment must usually be solved by eliminating the problem at that particular piece of equipment. Interference caused by nearby AC sources is caused by removal of the source or by shielding the ACI interface circuitry.

Strong RF fields near the interface circuitry can create problems. Remember that the Intercept Jr. components are not in shielded boxes and any operational situations where strong fields exist must be dealt with by shielding the circuitry.

Supply voltage transients are present when heavy loads are switched on and off. Also when weak batteries are supplying the power, internal battery resistance increases the possibility of transients. Power supply problems can be minimized by using a line operated power source and by decoupling high current loads. LEDs and TTY current loops are examples of high current loads.

6954 ACI BOARD

The 6954 board is designed to work with +5 volts and -5 volts. The +5 volts are supplied by the Intercept Jr. batteries and the -5 volts are generated by a voltage converter. The 6954 board should not be operated over a wide range of supply voltages. If batteries are being used and they are being heavily loaded, it may be necessary to use a line operated power supply to maintain a more constant +5 volt supply voltage. If the supply voltage drops too low, the phaselock loops (IC1 and IC2) may have to be readjusted to correctly receive data.

WRITING PROGRAMS FOR DATA TRANSFERS - USING TTY ADDRESSES

For each card in the system that uses a 6101 PIE chip, a unique address must be assigned to that card so that I/O instructions can be directed to a particular board. If a TTY interface is not being used, the address normally used for the TTY can be assigned to the ACI board. The advantage of doing this is that there are ROM programs that reference the TTY board address. The ROM programs allow you to initialize the PIE, write data to the recorder, and read data from the recorder simply by calling the existing programs from ROM. If a TTY card and an ACI are both being used, a different address must be assigned and the ROM routines cannot be used. A listing of the ROM is in the Intercept Jr. manual. The pertinent routines are listed below and can be used as a guide to writing your own program. Note that the I/O instructions that address the standard TTY card must be modified for other addresses.

```

                                / THE PIE INITIALIZE ROUTINE
INPIE,    CAF                    / CLEAR ALL FLAGS
          6400                   / CLEAR THE DISPLAY
          TAD KCRA                / GET THE CRA WORD
          WCRB                    / LOAD IT TO CRA IN PIE
          CLA CLL
          TAD KCRB                / GET THE CRB WORD
          WCRB                    / LOAD IT TO THE CRB WORD IN PIE
          CLA CLL
          TAD KTTY                / GET THE TTY-UART CONTROL WORD
          WRITE2                  / LOAD IT TO UART CONTROL WORD
          CLA CLL
          WVR                      / WRITE ALL ZEROS INTO THE VECTOR WORD
          DCA SAVS                / CLEAR SAVS
          RETURN                  / GO BACK TO THE PROGRAM

KCRA,    7200
KCRB,    1560
KTTY,    7600

                                / THE PRINT TO TTY ROUTINE
TALK,    SKIP2                   / SKIP ON CLEAR XMIT BUFFER
          JMP .-1                 / XMIT BUFFER NOT YET CLEAR
          WRITE1                  / WRITE THE AC TO THE UART
          DCA TIME                / CLEAR THE AC AND STORE THE DATA
                                / IN TIME FOR POSSIBLE RECOVERY
          RETURN                  / GO BACK TO THE PROGRAM

                                / LISN IS THE ROUTINE TO GET A
                                / CHARACTER FROM THE TTY KEYBOARD
                                / CR READER

READ,    SKIP1                   / WAIT FOR DATA READY FLAG
          JMP .-1
          CLA                      / CLEAR THE AC
          READ1                    / READ THE UART BUFFER AND ERROR
                                / FLAGS, CLEAR THE DATA READY FLAG
          AND TTYM                 / CLEAR OUT THE UNWANTED BITS
          RETURN                  / GO BACK TO THE PROGRAM

TTYM,    0377
```

USER ASSIGNED ADDRESS

The user should consult the 6101-PIE data sheet and thoroughly understand its operation in order to write effective data transfer programs.

I/O instructions are of the format

0	1	2	3	4	5	6	7	8	9	10	11
1	1	0	Address					Control			

Bits 3-7 are compared with the SEL3-SEL7 inputs of the PIE.

SEL3-SEL7 represent the PIE address to be selected. The standard teletype address is SEL3 and SEL4 zero and SEL5, SEL6 & SEL7 a high logic level -- so all TTY I/O instructions are of the form 11000111XXXX, where the X's represent the type of I/O instruction to be implemented. The following are examples of TTY I/O instructions: WRITE2 (OCTAL 6171), READ1 (OCTAL 6160), and SKIP2 (OCTAL 6163).

WRITING A MEMORY WORD TO THE CASSETTE

Assume that we set the PIE address as follows:

SEL3=0

SEL4=0

SEL5=0

SEL6=1

SEL7=1

The WRITE1 command is used to send an eight bit character out on the DX bus to the UART where it is transmitted as serial data. Due to our choice of PIE address, the octal code for WRITE1 will be 6061. In addition to the WRITE1 command, we must use a SKIP2 command to form a waiting loop - the loop is necessary since the UART may not be ready to transmit a character at all times. A program segment which will write an eight bit word is:

0020	6063	LOOP, SKIP2
0021	5020	JMP LOOP
0022	6061	WRITE1

The program will cycle between the first two instructions until the UART is ready to transmit a new character - at that time the program will skip the second instruction and go to the WRITE1 instruction. Note that only eight bits can be transferred and it will take two transfers to write a 12 bit memory word.

READING A MEMORY WORD FROM THE CASSETTE

The reading routine is very similar to the write routine and is shown below. Note that the four most significant bits are masked off.

0030	6062	LOOP, SKIP1
0031	5030	JMP LOOP
0032	7200	CLA
0033	6060	READ1
0034	0040	AND MASK
	.	
	.	
	.	
0040	0377	MASK, 0377

INITIALIZING THE PIE

At the start of a program, the internal registers of the PIE chip must be initialized. The ROM routine which does this has the label INPIE. If the standard TTY address is not used, the I/O instructions must be modified to conform to the address being used.

THEORY OF RECEIVER SECTION OPERATION

IC1 is a 565 phaselock loop that locks on to the mark and space frequencies from the cassette. The center frequency of the phaselock loop is controlled by C4, R5, and P1. The center frequency is preset at the factory with P1. Capacitor C5 is necessary to prevent oscillations. The signal from the tape recorder is AC coupled through C1 filter. Pins 6 and 7 are the 565 output terminals. Pin 6 is a reference DC voltage while pin 7 is a control voltage that goes above and below the reference voltage--depending on whether a mark or a space is being received. The control voltage passes through a low pass filter. The filtered control voltage goes to pin 4 of IC3.

IC3 is a precision comparator which compares the ICI reference output with IC1 control voltage. The output of the comparator at pin 9 is a one (5 volt) level when a mark frequency is being received and is a zero when a space is received.

IC2 is a 565 phaselock loop which acquires the recorded clock signal and tracks any variations in clock speed. The composite signal (clock and data) is applied to pin 2. C14 and R15 are set so that the PLL center frequency is equal to the normal clock frequency. The phaselock loop VCO output at pins 4 and 5 supply the clock signal to a UART. When the PLL is locked onto the clock, pins 4 and 5 represent recovered clock data.

The following test points are available in the receiver section:

TP1	MARK/SPACE CONTROL VOLTAGE
TP2	REFERENCE VOLTAGE
TP3	RECEIVED DIGITAL DATA
TP4	RECOVERED CLOCK SIGNAL

Figure 2 shows typical waveforms for the test points.

THEORY OF TRANSMITTER SECTION OPERATION

The transmitter section consists of IC4, IC5, IC8, IC9 and IC10.

The circuit consisting of IC9 and a crystal oscillator is the main time base from which the transmitted clock is derived and from which the mark and space frequencies are derived. The oscillator output is at pin 5 at a frequency of 2.4576 MHz. The oscillator signal is fed to pin 10 of IC10. IC10 is a multistage divider chip that provides three different frequencies -- 19200 Hz and 9600 Hz for the mark/space generation and 4800 Hz for the UART transmit clock. The 4800 Hz signal is also supplied to a mixer (R18 and R22) and eventually recorded on the tape. The circuitry of IC5 is a multiplexer which either sends 19200 Hz or 9600 Hz to the digital sinewave generator (IC4).

The counter, IC4, is used to digitally generate the mark/space sinewave frequency. The outputs of IC4 are pins 3, 6 and 11. R19, R20, R21 and R22 comprise a mixer circuit -- as the counter outputs go high, they are mixed with the composite signal across R22. Also appearing at R22 will be the clock signal. The composite signal passes through an active filter (Q2 and associated components). The output of the filter is divided down to an optimum record signal and sent to the recorder through the "MIC" plug. Optional jumpers are available to accomodate special recorder requirements.

The following test points are available in the transmitter section:

TP5	Composite Sinewave
TP7	19200 Hz Digital Signal
TP8	4800 Hz Digital Signal
TP9	9600 Hz Digital Singal

Figure 3 shows typical waveshapes. Note that the signal to the recorder is a low level signal intended for the microphone input, not the auxiliary input. Some recorders have an "AUX" input, but all recorders have a "MIC" input.

D.C.-D.C. CONVERTER

IC13 is a Texas Instrument TL497 switching converter. IC13 switches current into L1 at a frequency determined by C30. As the current through L1 is switched off, a negative voltage is present at pin 8 of IC13. CR1 passes this negative pulse to C31 where it is filtered and sent to the circuits requiring -5 volts. The voltage at the output is set by the resistor ratio of R32 and R33. R34 is used as a current limit and to reduce noise on the +5 volt line.

CIRCUIT OPTIONS

The 6954-ACI board allows the user to implement a remote power ON/OFF control to a recorder if desired. PIE flag F1 is used via a gate and transistor to operate a DIP relay. The contacts are brought to the plug bracket. An extra hole is provided on this bracket for a 2.5 mm plug. One way to use this feature is to parallel the remote foot-switch or PAUSE switch available on many recorders. Another possibility is to shut off all power to the system on a user program decision. Optional resistor locations and jumper pads have been provided in the filter output circuit to adjust voltage levels. In most cases, the factory settings should be adequate.

APPENDIX A
INTERCEPT JR. PROGRAMMING FUNDAMENTALS

NUMBER SYSTEMS

INTERCEPT JR., as most digital computers, uses the binary system. Representation of binary numbers by positional notation is analogous to the common representation of decimal numbers by assigning ten different "weights" to each position. Any number of n digits may be written as the string of digits.

$$C_{n-1} C_{n-2} \dots C_1 C_0$$

where C's can range from 0 to 9.

This actually stands for

$$\begin{aligned} &C_{n-1} \text{ followed by } (n-1) \text{ zeros} + \\ &C_{n-2} \text{ followed by } (n-2) \text{ zeros} + \\ &\vdots \\ &C_1 \text{ followed by } 1 \text{ zero} + \\ &C_0 \end{aligned}$$

$$\text{or } C_{n-1} (10)^{n-1} + C_{n-2} (10)^{n-2} + \dots C_1 (10)^1 + C_0 (10)^0$$

For example, 1234 is $1000 + 200 + 30 + 4$ or $1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4$.

Similarly, in the binary system, any number may be represented by a string of coefficients

$$B_{n-1} B_{n-2} \dots B_0 B_1$$

which stands for

$$B_{n-1} (2)^{n-1} + B_{n-2} (2)^{n-2} + B_1 (2)^1 + B_0 (2)^0$$

where the B's may be 0 or 1.

The "radix", or base of the binary system is 2, whereas it is 10 in the case of the decimal system.

In theory, binary numbers may be used to describe the condition of these flip-flops.

A system with 12 flip-flops could be represented by a 12 bit number, for example 1 0 1 1 0 0 1 1 1 0 0 1, where each bit represents the set or reset state of a particular flip-flop. Binary numbers are unwieldy to handle because of the long strings involved, so often a simplification is introduced.

Consider the numbers 0 through 15 written in their binary equivalent.

2^3	2^2	2^1	2^0	
8	4	2	1	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Observe that in the "units", or 2^0 position, the state changes, or "toggles" most often, for example every time the number increments. In the next or 2^1 position, the bits toggle every two increments, and in the 2^2 position, every four times, etc.

Or, looking at this another way, the one bit groups 0 and 1 alternate every time, the two bit groups 00, 01, 10, 11 recur every fourth time, the three bit groups

```

000 )
001 )
010 )
011 )
100 )
101 )
110 )
111 )
```

recur every eight times, and so on.

Thus, to shorten binary numbers, we could encode these groups. The tradeoff is between the length of the number string, and the number of symbols required.

Our 12 bit number may now be represented by three of the above codes:

	1011	0011	1001
or	B	3	9

So, we have doubled the number of symbols to sixteen but reduced the length of our string only by one, from four to three. The code itself has also become a little unwieldy because the number of different symbols.

As a matter of fact, this representation by four bit groups is known as the hexadecimal system (base 16 system) and is widely used.

The system of representation with three bit groups encoded with the eight symbols 0 through 7 is known as the octal number system and is also in wide use.

We shall adopt the octal numbering system for INTERCEPT JR.

It should be evident by now that the choice is based purely on convenience and consistency with the available literature as almost all digital computers are fundamentally binary machines.

At this point, it is instructive to turn your machine ON. Press the CNTRL key and the MEM key and then keep pressing the MEM key. The address display will increment in an octal progression. By watching the addresses increment, the user can become familiar with the octal system.

To recapitulate, conversion from binary to octal is done by taking groups of three bits, starting from the least significant bit, filling in a zero or zeros to the most significant group, if necessary, and writing down the octal equivalent for each group.

Conversion from octal to binary is done by directly writing down three bits from each octal number.

INTERCEPT JR. uses two's complement arithmetic in its processing logic.

The processor performs binary addition between two operands but binary subtraction is best done adding the "negative" of one operand to the other. This requires an extra symbol to indicate the sign of the number. To avoid this, a form of representation known as two's complement has been devised to represent negative numbers.

APPENDIX B
INTRODUCTION TO LOGIC

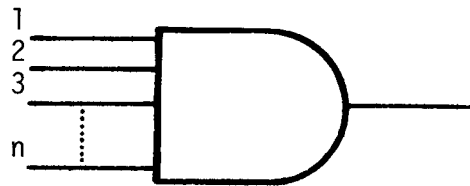
INTRODUCTION

This appendix briefly reviews truth tables as applied to simple logic elements, both combinatorial and sequential. Timing diagrams and state diagrams are illustrated using flip-flops as examples.

TRUTH TABLES

AND FUNCTION

Symbol for AND gate

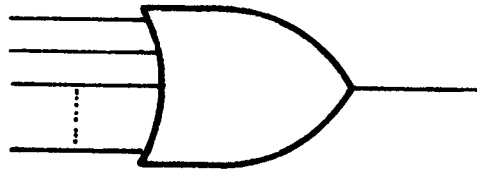


Output is true only if all inputs are true, that is, input 1 AND input 2 AND...AND input N

Input 1	Input 2	Input N	Output
0	1	1	0
0	0	1	0
0	0	0	0
1	0	1	0
1	0	0	0
		⋮		
1	1	all 1's	1	1

This table shows a conventional positive logic AND gate, with 1 representing logic high or true, and 0 representing logic low or false. Thus, only one combination of the inputs gives a high output.

OR FUNCTION SYMBOL FOR OR GATE



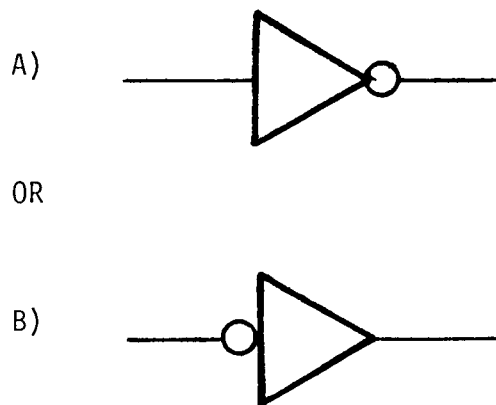
Output is true if at least one of the inputs is true, for example Input 1 OR Input 2 OR...Input N OR any combination of true inputs yields a true output.

Input 1	Input 2	Input N	Output
0	0	all 0's	0	0
0	1	0	1
1	1	0	1
0	0	1	1

Here, only one of the 2^N possible input combinations namely all 0's will yield a false or low output.

NOT FUNCTION

Symbol for inverter



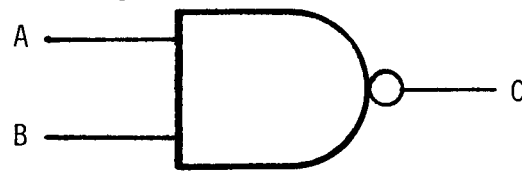
Output is logical inversion of input.

Input	Output
1	0
0	1

The position of the "bubble" tells you what the active level of the input is expected to be by the designer. Quite often, it is drawn as in A above.

NAND FUNCTION

Symbol for NAND gate

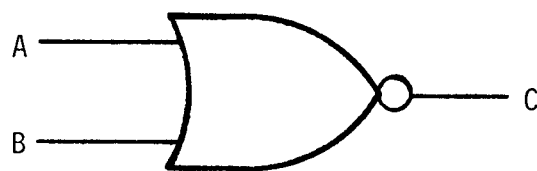


A	B	C
1	1	0
0	1	1
1	0	1
0	0	1

This is the same as an AND gate followed by an inverter.

NOR FUNCTION

Symbol for NOR gate

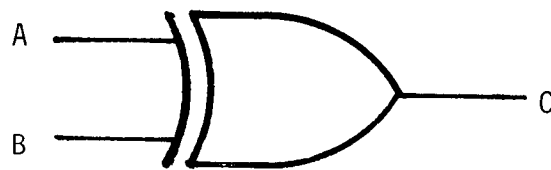


A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

This is the same as an OR gate followed by an inverter.

EXCLUSIVE-OR FUNCTION

Symbol for EX-OR gate

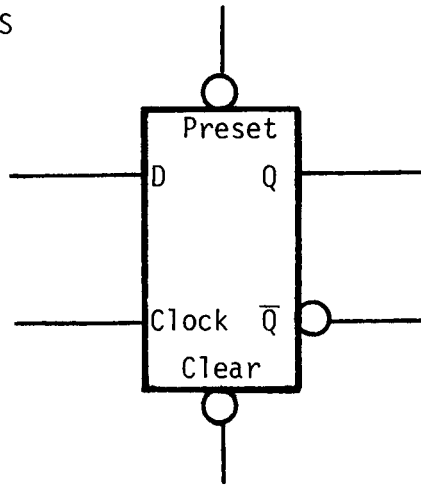


A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Output is true if Input A OR B but not BOTH are true. Note that this gate can be used to detect the fact that the inputs are identical. Thus, it is used quite often in digital comparators.

D-TYPE FLIP-FLOPS

Symbol



TRUTH TABLE

D	<u>Input</u>			<u>Output</u>	
	Clock	Clear	Preset	Q	\bar{Q}
X	X	0	0	1	1
X	X	0	1	0	1
X	X	1	0	1	0
X	0	1	1	STABLE	
X	1	1	1	STABLE	
X	↓	1	1	STABLE	
0	↑	1	1	0	1
1	↑	1	1	1	0

The truth table for a D flip-flop is complicated by the sequential nature of this logic device. Strictly speaking, truth tables should represent combinatorial logic properties only.

In this case, the truth table also shows the edge-triggered action of the flip-flop with ↓ representing the negative going edge and ↑ the positive going edge. 0 and 1 show stable levels.

The table is really a hybrid of a combinatorial truth table and a state table.

This flip-flop is a synchronous storage element. In other words, it stores data using a clock signal to synchronize the operation. In this case, the device is positive-going edge triggered, or simply, positive edge triggered.

The bottom two lines show that as long as the clear and present inputs are high, the positive clock edge loads the flip-flop with the data at D such that the Q output reflects the D input. The \bar{Q} output is always supposed to be the inverse of the Q input.

All other conditions of the clock--high, low, or negative edge, have no effect and the outputs remain stable (at the value loaded on the previous positive edge).

The D flip-flop thus delays data by one clock period. Note that during the preceding discussion, the clear and preset inputs were assumed high.

These inputs are asynchronous, and so can change the outputs regardless of the clock or data input.

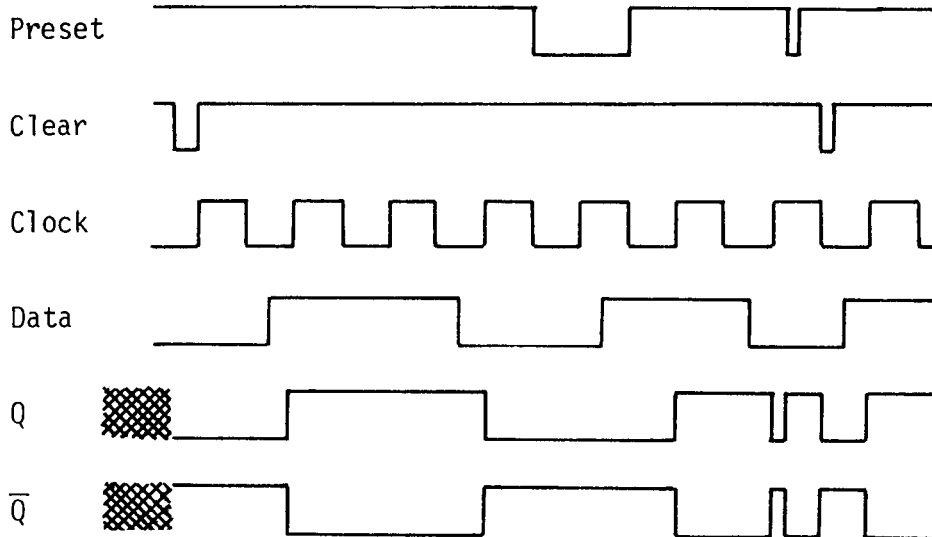
The bubbles indicate active low operation.

When both asynchronous, or "direct" inputs are low, both Q and \bar{Q} go high, so this condition is normally forbidden.

In sequential circuits, other time related parameters are generally specified. Thus data inputs generally have to meet setup and hold times with respect to the active edge of the clock, or "interrogating" edge. A setup time is the time the data must be present before the active edge, and the hold time is the time for which it must continue to be present--"held", after the active edge in order for proper operation. Sequential device operation can be much better understood using another graphical technique known as a timing diagram. Such diagrams bring out the time-sequential interactions in these devices much more clearly. The next section will deal with timing diagrams.

TIMING DIAGRAMS

Shown below is a timing diagram for a D flip-flop.



STATE DIAGRAMS

Sequential circuits inherently contain storage elements each of which may be in one of two stable states. Each "state" of a digital system, as explained in the section on truth tables, could be represented by a binary number. The system changes states in response to internal and/or external conditions. The state transition may be synchronous to a clock pulse train or asynchronous. Asynchronous sequential circuits will not be covered in detail in this book, and we shall deal only with clocked logic.

State tables and state transition diagrams are additional tools of analysis and design that digital engineers use.

As an example, we shall show the state table and state transition diagram for the J-K flip-flop.

Q_n	J_n	K_n	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

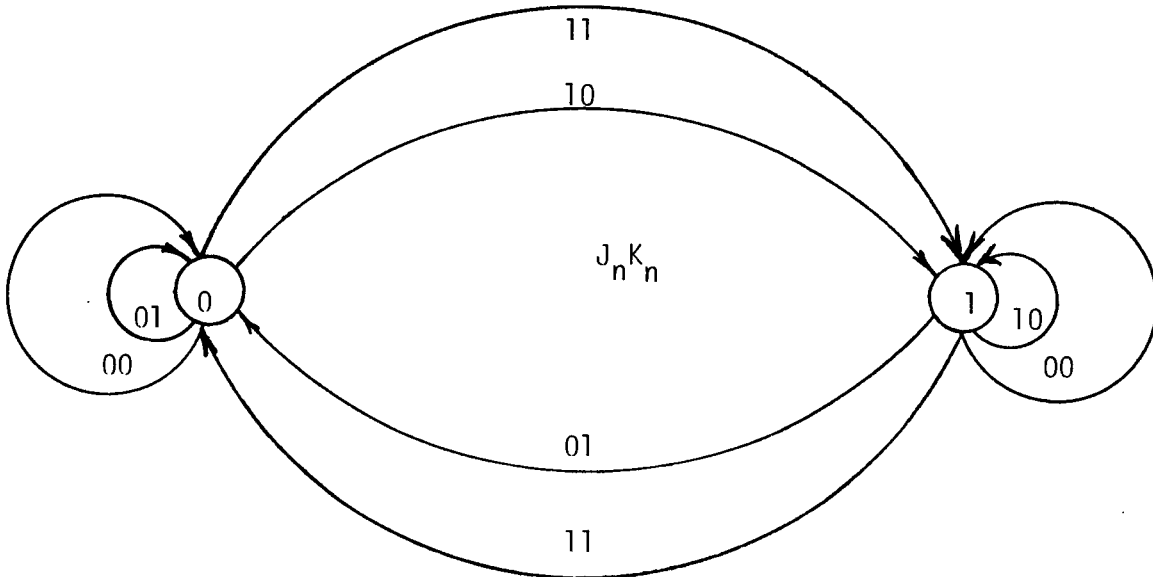
The transition, if any, from Q_n to Q_{n+1} (Q at time t_n and Q at time t_{n+1}) is triggered by the negative going edge of the clock.

In words, when J and K are zero, the outputs do not change. When J and K are both one, the output toggles at every clock pulse and when J and K are at opposite levels, Q follows J (and \bar{Q} follows K).

Another form of the state table shows this relationship:

	$J_n=0, K_n=0$	$J_n=0, K_n=1$	$J_n=1, K_n=0$	$J_n=1, K_n=1$
Present State	Next State	Next State	Next State	Next State
Q_n	Q_{n+1}	Q_{n+1}	Q_{n+1}	Q_{n+1}
0	0	0	1	1
1	1	0	1	0

The number of inputs and outputs in a digital system are not related to the number of states. They only determine the number of paths along which a change of state may occur. In this specific case, the output is also the state.



The state diagram shows the different states of a digital system and the conditions necessary to cause the system to change states.

Information that is not shown on a state transition diagram is presented in other visual aids such as timing diagrams.

Thus, in general, a complex system must be studied with the aid of many different tools in order to gain insight into the operation of the system from many different angles.

Digital systems may be "hardwired" or programmable. Hardwired digital systems have many logic devices scattered at random and many operations are done in parallel.

This "random logic" consists of such standard SSI and MSI functions as counters, multiplexers, decoders, latches, registers, etc.

Programmable logic systems usually have denser, more regularly arrayed chips such as ROMs, PROMs, RAMs, FPLAs, microprocessors, etc. and substitute many sequential operations for a single parallel operation, though this is not always the case.

Such systems replace the "randomness" in the logic with random bit patterns in the memory components. Programmable logic systems are gaining popularity with the advent of inexpensive LSI storage and processor devices.

3000 to 3777 (Octal) to 1836 to 2047 (Decimal)

Table with 8 columns (3000-3079) and 8 rows (0-7) of octal to decimal conversion data.

4000 to 4777 (Octal) to 2048 to 2599 (Decimal)

Table with 8 columns (4000-4079) and 8 rows (0-7) of octal to decimal conversion data.

5000 to 5777 (Octal) to 2600 to 3071 (Decimal)

Table with 8 columns (5000-5079) and 8 rows (0-7) of octal to decimal conversion data.

6000 to 6777 (Octal) to 3072 to 3583 (Decimal)

Table with 8 columns (6000-6079) and 8 rows (0-7) of octal to decimal conversion data.

7000 to 7777 (Octal) to 3584 to 4095 (Decimal)

Table with 8 columns (7000-7079) and 8 rows (0-7) of octal to decimal conversion data.

8000 to 8777 (Octal) to 4096 to 4607 (Decimal)

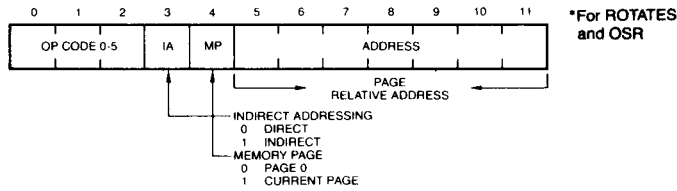
Table with 8 columns (8000-8079) and 8 rows (0-7) of octal to decimal conversion data.

APPENDIX D
INSTRUCTION SUMMARY
AND
BIT ASSIGNMENTS

BASIC INSTRUCTIONS

MNEMONIC	OCTAL CODE	OPERATION	NO. OF STATES		
			DIR	IND	AUTO
AND	0000	Logical AND	10	15	16
TAD	1000	Binary ADD	10	15	16
ISZ	2000	Increment, and skip if zero	16	21	22
DCA	3000	Deposit and clear AC	11	16	17
JMS	4000	Jump to subroutine	11	16	17
JMP	5000	Jump	10	15	16
IOT	6000	In/out transfer	17	—	—
OPR	7000	Operate	10/15*	—	—

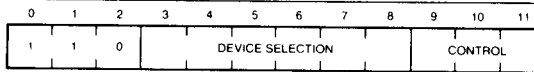
MEMORY REFERENCE INSTRUCTION FORMAT



PROCESSOR IOT INSTRUCTIONS

MNEMONIC	OCTAL CODE	OPERATION	NO. OF STATES
SKON	6000	Skip if interruption on	17
ION	6001	Interrupt turn on	17
IOF	6002	Interrupt turn off	17
SRQ	6003	Skip if INT request	17
GTF	6004	Get flags	17
RTF	6005	Return flags	17
SGT	6006	Operation is determined by external devices, if any	17
CAF	6007	Clear all flags	17

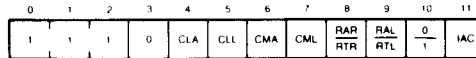
BIT ASSIGNMENTS IOT



GROUP 1 OPERATE MICROINSTRUCTIONS

MNEMONIC	OCTAL CODE	OPERATION	LOG SEQ.	NO. OF STATES
NOP	7000	No operation	1	10
IAC	7001	Increment accumulator	3	10
RAL	7004	Rotate accumulator left	4	15
RTL	7006	Rotate two left	4	15
RAR	7010	Rotate accumulator right	4	15
RTR	7012	Rotate two right	4	15
BSW	7002	Byte swap	4	15
CML	7020	Complement link	2	10
CMA	7040	Complement accumulator	2	10
CIA	7041	Complement and increment accumulator	2,3	10
CLL	7100	Clear link	1	10
CLL RAL	7104	Clear link—rotate accum. left	1,4	15
CLL RTL	7106	Clear link—rotate two left	1,4	15
CLL RAR	7110	Clear link—rotate accum. right	1,4	15
CLL RTR	7112	Clear link—rotate two right	1,4	15
STL	7120	Set the link	1,2	10
CLA	7200	Clear accumulator	1	10
CLA IAC	7201	Clear accumulator—Increment accumulator	1,3	10
GLT	7204	Get the link	1,4	15
CLA CLL	7300	Clear accumulator—clear link	1	10
STA	7240	Set the accumulator	1,2	10

BIT ASSIGNMENTS GROUP 1



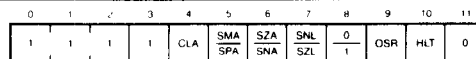
BSW IF BITS
8 & 9 ARE 0
AND BIT 10 IS 1

LOGICAL SEQUENCES
1—CLA CLL
2—CMA CML
3—IAC
4—RAR RAL RTR RTL BSW

GROUP 2 OPERATE MICROINSTRUCTIONS

MNEMONIC	OCTAL CODE	OPERATION	LOG SEQ.	NO. OF STATES
NOP	7400	No operation	1	10
HLT	7402	Halt	3	10
OSR	7404	Or with switch register	3	15
SKP	7410	Skip	1	10
SNL	7420	Skip on non-zero link	1	10
SZL	7430	Skip on zero link	1	10
SZA	7440	Skip on zero accumulator	1	10
SNA	7450	Skip on non-zero accumulator	1	10
SZA SNL	7460	Skip on zero accum. or skip on non-zero link, or both	1	10
SNA SZL	7470	Skip on non-zero accum. and skip on zero link	1	10
SMA	7500	Skip on minus accumulator	1	10
SPA	7510	Skip on positive accumulator	1	10
SMA SNL	7520	Skip on minus accum. or skip on non-zero link or both	1	10
SPA SZL	7530	Skip on positive accum. and skip on zero link	1	10
SMA SZA	7540	Skip on minus accum. or skip on zero accum. or both	1	10
SPA SNA	7550	Skip on positive accum. and skip on non-zero accum.	1	10
SMA SZA SNL	7560	Skip on minus accum. or skip on zero accum. or skip on non-zero link or all	1	10
SPA SNA SZL	7570	Skip on positive accum. and skip on non-zero accum. and skip on zero link	1	10
CLA	7600	Clear accumulator	2	10
LAS	7604	Load accumulator with switch register	1,3	15
SZA CLA	7640	Skip on zero accum. then clear accum.	1,2	10
SNA CLA	7650	Skip on non-zero accum. then clear accumulator	1,2	10
SMA CLA	7700	Skip on minus accum. then clear accumulator	1,2	10
SPA CLA	7710	Skip on positive accum. then clear accumulator	1,2	10

BIT ASSIGNMENTS GROUP 2

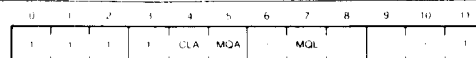


LOGICAL SEQUENCES
1 (Bit 8 is Zero) SMA or SZA or SNL
(Bit 8 is One) SPA and SNA and SZL
2 CLA
3 OSR HLT

GROUP 3 OPERATE MICROINSTRUCTIONS

MNEMONIC	OCTAL CODE	OPERATION	LOG SEQ.	NO. OF STATES
NOP	7401	No operation	3	10
MQL	7421	MQ register load	2	10
MQA	7501	MQ register into accumulator	2	10
SWP	7521	Swap accum. and MQ register	3	10
CLA	7601	Clear accumulator	1	10
CAM	7621	Clear accum. and MQ register	3	10
ACL	7701	Clear accum. and load MQ register into accumulator	3	10
CLA SWP	7721	Clear accum. and swap accum. and MQ register	3	10

BIT ASSIGNMENTS GROUP 3



LOGICAL SEQUENCE
1—CLA
2—MQA MQL
3—ALL OTHERS
Don't Care

APPENDIX E

GLOSSARY

ABSOLUTE ADDRESS: A binary number that is permanently assigned as the address of a memory storage location.

ACCESS TIME: The time required to locate an off-line storage location.

ACCESSING DATA: The process of locating the off-line storage location with which data is to be transferred.

ACCUMULATOR: A 12-bit register in which the result of an operation is formed; abbreviation: AC.

ADDRESS: A label, name, or number which designates a location where information is stored.

ADDRESSING: The term given to the act of selecting a word in memory.

ALGORITHM: A prescribed set of well-defined rules or processes for the solution of a problem in a finite number of steps.

ALPHANUMERIC: Pertaining to a character set that contains both letters and numerals, and usually other characters.

ARGUMENT:

1. A variable or constant which is given in the call of a subroutine as information to it.
2. A variable upon whose value the value of a function depends.
3. The known reference factor necessary to find an item in a table or array (i.e. the index).

ARITHMETIC AND LOGIC UNIT (ALU): The unit which performs both arithmetic and logic operations.

ARITHMETIC UNIT: The component of a computer where arithmetic and logical operations are performed.

ASCII: An abbreviation for American Standard Code for Information Interchange.

ASSEMBLE: To translate from a symbolic program to a binary program by substituting binary operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses.

ASSEMBLER: A program which translates symbolic op-codes into machine language and assigns memory locations for variables and constants.

AUTO-INDEXING: When one of the absolute locations from 0010 through 0017 is addressed indirectly, the content of that location is incremented by one, rewritten in that same location, and used as the effective address of the current instruction.

AUXILLARY STORAGE: Storage that supplements memory such as disk or tape.

BASE ADDRESS: A given address from which an absolute address is derived by combination with a relative address, synonymous with address constant.

BINARY: Pertaining to the number of system with a radix of two.

BINARY CODE: A code that makes use of exactly two distinct characters, 0 and 1.

BIT: A binary digit. In the IM6100 microprocessor each word is composed of 12 bits.

BLOCK: A set of consecutive machine words, characters, or digits handled as a unit, particularly with reference to I/O.

BOOTSTRAP: A technique or device designed to bring a program into the computer from an input device.

BRANCH: A point in a routine where one of two or more choices is made under control of the routine.

BUFFER: A storage area.

BUG: A mistake in the design or implementation of a program resulting in erroneous results.

BYTE: A group of binary digits usually operated upon as a unit.

CALL: To transfer control to a specified routine.

CALLING SEQUENCE: A specified set of instructions and data necessary to set up and call a given routine.

CENTRAL PROCESSING UNIT: The unit of a computing system that includes the circuits controlling the interpretation and execution of instructions--the computer proper, excluding I/O and other peripheral devices.

CHARACTER: A single letter, numeral, or symbol used to represent information.

CLEAR: To erase the contents of a storage location by replacing the contents, normally with zeros or spaces; to set to zero.

CODING: To write instructions for a computer using symbols meaningful to the computer, or to an assembler, compiler or other language processor.

COMMAND: A user order to a computer system, usually given through a Teletype keyboard.

COMMAND DECODER: That part of a computer system which interprets used commands. Also called command-string decoder.

COMPATIBILITY: The ability of an instruction or source language to be used on more than one computer.

COMPILE: To produce a binary-coded program from a program written in source (symbolic) language, by selecting appropriate subroutines from a subroutine library, as directed by the instructions or other symbols of the source program. The linkage is supplied for combining the subroutines into a workable program, and the subroutine and linkage are translated into binary code.

COMPILER: A program which translates statements and formulas written in a source language into a machine language program, e.g. a FORTRAN Compiler. Usually generates more than one machine instruction for each statement.

COMPLEMENT: (One's) To replace all 0 bits with 1 bits and vice versa. (Two's) To form the one's complement and add 1.

CONDITIONAL ASSEMBLY: Assembly of certain parts of a symbolic program only if certain conditions have been met.

CONDITIONAL SKIP: Depending upon whether a condition within the program is met, control may transfer to another point in the program.

CONSOLE: Usually the external front side of a device where controls and indicators are available for manual operation of the device.

CONVERT:

1. To change numerical data from one radix to another.
2. To transfer data from one recorded format to another.

CORE MEMORY: The main high-speed storage of a computer in which binary data is represented by the switching polarity of magnetic cores.

COUNT: The successive increase or decrease of a cumulative total of the number of times an event occurs.

COUNTER: A register or storage location (variable) used to represent the number of occurrences of an operation.

CURRENT LOCATION COUNTER: A counter kept by an assembler to determine the address assigned to an instruction or constant being assembled.

CURRENT PAGE: The page of memory "pointed to" or addressed by the Program Counter. The page we are on.

CYCLE TIME: The length of time it takes the computer to reference one word of memory.

DATA: A general term used to denote any or all facts, numbers, letters and symbols. It connotes basic elements of information which can be processed or produced by a computer.

DATA BREAK: A facility which permits I/O transfers to occur on a cycle-stealing basis without disturbing program execution.

DEBUG: To detect, locate and correct mistakes in a program.

DEVICE FLAGS: One-bit registers which record the current status of a device.

DIGITAL COMPUTER: A device that operates on discrete data, performing sequences of arithmetic and logical operations on this data.

DIRECT ADDRESS: An address that specifies the location of an instruction operand.

DOUBLE PRECISION: Pertaining to the use of two computer words to represent one number. In the IM6100 a double precision result is stored in 24 bits.

DUMP: To copy the contents of all or part of core memory, usually onto an external storage medium.

EFFECTIVE ADDRESS: The address actually used in the execution of a computer instruction.

EXECUTE: To carry out an instruction or run a program on the computer.

EXTERNAL STORAGE: A separate facility or device on which data usable by the computer is stored (such as paper tape, tape or disk).

FIELD:

1. One or more characters treated as a unit.
2. A specified area of a record used for a single type of data.
3. A division of memory on a IM6100 computer referring to a 4K section of core.

FILE: A collection of related records treated as a unit.

FLAG: A variable or register used to record the status of a program or device. In the latter case, also called a device flag.

FLIP-FLOP: A device with two stable states.

FLOATING POINT: A number system in which the position of the radix point is indicated by one part of the number (the exponent) and another part represents the significant digits (the mantissa), I/O.

FLOWCHART: A graphical representation of the operations required to carry out a data processing operation.

HARDWARE: Physical equipment, e.g., mechanical, electrical or electronic devices.

HEAD: A component that reads, records or erases data on a storage device.

INDIRECT ADDRESS: An address in a computer instruction which indicates a location where the address of the referenced operand is to be found.

INITIALIZE: To set counters, switches, and addresses to zero or other starting values at the beginning of, or at pre-scribed points in, a computer routine.

INSTRUCTION: A command which causes the computer or system to perform an operation. Usually one line of a source program.

INSTRUCTION FETCH (IFETCH): The act of completing an instruction address to memory and returning to the Microprocessor with the instruction.

INSTRUCTION REGISTER (IR): The register which holds the instruction when it is obtained, or received, from memory.

INTERNAL STORAGE: The storage facilities forming an integral physical part of the computer and directly controlled by the computer. Also called main memory.

INTERPRETER: A program that translates and executes source language statements at run time.

I/O: Abbreviation for input/output.

JOB: A unit of code which solves a problem, i.e. a program and all its related subroutines and data.

JUMP: A departure from the normal sequence of executing instructions in a computer.

K: An abbreviation for the prefix kilo, i.e. 1000 in decimal notation.

LABEL: One or more characters used to identify a source language statement or line.

LANGUAGE, ASSEMBLY: The machine-oriented programming language used by an assembly system.

LANGUAGE, COMPUTER: A systematic means of communicating instructions and information to the computer.

LANGUAGE, MACHINE: Information that can be directly processed by the computer, expressed in binary notation.

LANGUAGE, SOURCE: A computer language such as PAL III or FOCAL in which programs are written and which require extensive translation in order to be executed by the computer.

LEADER: The blank section of tape at the beginning of the tape.

LEAST SIGNIFICANT DIGIT: The right-most digit of a number.

LIBRARY ROUTINES: A collection of standard routines which can be incorporated into larger programs.

LINE FEED: The Teletype operation which advances the paper by one line.

LINE NUMBER: In source languages such as FOCAL, BASIC, and FORTRAN, a number which begins a line of the source program for purposes of identification. A numeric label.

LINK:

1. A one-bit register in the IM6100.
2. An address pointer generated automatically by the PAL-D or MACRO-8 Assembler to indirectly address an off-page symbol.
3. An address pointer to the next element of a list, or the next block number of a file.

LIST:

1. A set of items.
2. To print out a listing on the line printer or Teletype.

LOAD: To place data into internal storage.

LOCATION: A place in storage or memory where a unit of data or an instruction may be stored.

LOOP: A sequence of instructions that is executed repeatedly until a terminal condition prevails.

MACHINE LANGUAGE PROGRAMMING: In this text, synonymous with assembly language programming. This term is also used to mean the actual binary machine instructions.

MACRO INSTRUCTION: An instruction in a source language that is equivalent to a specified sequence of machine instructions.

MANUAL INPUT: The entry of data by hand into a device at the time of processing.

MANUAL OPERATION: The processing of data in a system by direct manual techniques.

MASK: A bit pattern which selects those bits from a word of data which are to be used in some subsequent operation.

MASS STORAGE: Pertaining to a device such as disk or tape which stores large amounts of data readily accessible to the central processing unit.

MATRIX: A rectangular array of elements. Any table can be considered a matrix.

MEMORY:

1. The alterable storage in a computer.
2. Pertaining to a device in which data can be stored and from which it can be retrieved.

MEMORY ADDRESS REGISTER (MAR): The register which contains the address where information is to be read from memory or written (stored) into memory.

MEMORY PAGING: A system by which a memory is subdivided in order to permit addressing with a limited number of binary bits.

MEMORY PROTECTION: A method of preventing the contents of some part of main memory from being destroyed or altered.

MICROCOMPUTER: A complete small computing system that usually sells for less than \$5,000 and whose main processor building blocks are made of semiconductor integrated circuits. In function and structure it is similar to a minicomputer, with the main difference being price, size, speed and computing power.

MICROPROCESSOR: The semiconductor central processing unit (CPU) and one of the principal components of the microcomputer. The elements of the microprocessor are frequently contained on a single chip or within the same package but sometimes distributed over several chips. Microprocessors can contain registers, an arithmetic logic unit, a PLA, and associated timing and control logic.

MINICOMPUTER: A computer whose main frame sells for less than \$25,000. Usually it is a parallel binary system with 8, 12, 16, 18, or 24-bit word lengths incorporating semiconductor or magnetic memory offering 4K words to 32K words of storage. A naked minicomputer is one without cabinet, console and power supplies and consists of as little as a single PC card selling for less than \$1,000.

MONITOR: The master control program that observes, supervises, controls or verifies the operation of a system.

MQ REGISTER: A register which is program accessible and interacts with the Accumulator.

NESTING:

1. Including a program loop inside loop. Special rules apply to the nesting of FORTRAN DO-loops.
2. Algebraic nesting, such as $(A+B*(C+D))$, where execution proceeds from the innermost to the outermost level.

NORMALIZE: To adjust the exponent and mantissa of a floating-point number so that the mantissa appears in a prescribed format.

OBJECT PROGRAM: The binary coded program which is the output after translation of a source language program.

OCTAL: Pertaining to the number system with a radix of eight.

OFF-LINE: Pertaining to equipment or devices not under direct control of the computer, or processes performed on such devices.

ON-LINE: Pertaining to equipment or devices under direct control of the computer and to programs which respond directly and immediately to user commands.

OPERAND:

1. A quantity which is affected, manipulated or operated upon.
2. The address, or symbolic name, portion of an assembly language instruction.

OPERATOR: The symbol or code which indicates an action (or operation) to be performed, e.g. + or TAD.

OR: (Inclusive) A logical operation such that the result is true if either or both operands are true, and false if both operands are false. (Exclusive) A logical operation such that the result is true if either operand is true, and false if either or both operands are false. When neither case is specifically indicated, Inclusive OR is assumed.

ORIGIN: The absolute address of the beginning of a section of code.

OUTPUT: Information transferred from the internal storage of a computer to output devices or external storage.

OVERFLOW: A condition that occurs when a mathematical operation yields a result whose magnitude is larger than the program is capable of handling.

PAGE: A 128-word section of IM6100 memory beginning at an address which is a multiple of 200.

PASS: One complete cycle during which a body of data is processed. An assembler usually requires two passes during which a source program is translated into binary code.

PATCH: To modify a routine in a rough or expedient way.

PERIPHERAL EQUIPMENT: In a data processing system, any unit of equipment distinct from the central processing unit which may provide the system with outside storage or communication.

POINTER ADDRESS: Address of a memory location containing the actual (effective) address of desired data.

PRIORITY INTERRUPT: An interrupt which is given preference over other interrupts within the system.

PROCEDURE: The course of action taken for the solution of a problem.

PROGRAM COUNTER (PC): The register which contains, at any given time, the address in memory of the next instruction.

PROGRAMMED LOGIC ARRAY (PLA): That section of the Microprocessor which correctly sequences the Microprocessor for the appropriate instruction.

PSEUDO-OP: See Pseudo-operation.

PSEUDO-OPERATION: An instruction to the assembler; an operation code that is not part of the computer's hardware command repertoire.

PUSHDOWN LIST: A list that is constructed and maintained so that the next item to be retrieved is the item most recently stored in the list.

QUEUE: A waiting list. In time-sharing, the monitor maintains a queue of user programs waiting for processing time.

RADIX: The base of a number system; the number of digits symbols required by a number system.

RANDOM ACCESS: A storage device in which the addressability of data is effectively independent of the location of the data. Synonymous with direct access.

RANDOM ACCESS MEMORY: A memory whose content can be predetermined, stored indefinitely, changed at will and retrieved at random

READ ONLY MEMORY: A memory whose content, once predetermined, is permanent and can not be changed.

REAL-TIME: Pertaining to computation performed while the related physical process is taking place so that results of the computation can be used in guiding the physical process.

RECORD: A collection of related items of data treated as a unit.

RECURSIVE SUBROUTINE: A subroutine capable of calling itself.

REGISTER: A device capable of storing a specified amount of data, usually one word.

RELATIVE ADDRESS: The number that specified the difference between the actual address and a base address.

RELOCATABLE: Used to describe a routine whose instructions are written so that they can be located and executed in different parts of core memory.

RESPONSE TIME: Time between initialing an operation from a remote terminal and obtaining the result. Includes transmission time to and from the computer, processing time and access time for files employed.

RESTART: To resume execution of a program.

ROUTINE: A set of instructions arranged in proper sequence to cause the computer to perform a desired task. A program or subprogram.

RUN: A single, continuous execution of a program.

SEGMENT:

1. That part of a long program which may be resident in memory at any one time.
2. To divide a program into two or more segments or to store part of a routine on an external storage device to be brought into core as needed.

SERIAL ACCESS: Pertaining to the sequential or consecutive transmission of data to or from memory, as with paper tape: contrast with random access.

SHIFT: A movement of bits to the left or right frequently performed in the accumulator.

SIMULATE: To represent the function of a device, system or program with another device, system or program.

SINGLE STEP: Operation of a computer in such a manner that only one instruction is executed each time the computer is started.

SOFTWARE: The collection of programs and routines associated with a computer.

SOURCE LANGUAGE: See Language, source.

SOURCE PROGRAM: A computer program written in a source language.

STATEMENT: An expression or instruction in source language.

STORAGE ALLOCATION: The assignment of blocks of data and instructions to specified blocks of storage.

STORAGE CAPACITY: The amount of data that can be contained in a storage device.

STORAGE DEVICE: A device in which data can be entered, retained and retrieved.

STORE: To enter data into a storage device.

STRING: A connected sequence of entities such as characters in a command string.

SUBROUTINE, CLOSED: A subroutine not stored in the main part of a program, such a subroutine is normally called or entered with a JMS instruction and provision is made to return control to the main routine at the end of the subroutine.

SUBROUTINE, OPEN: A subroutine that must be relocated and inserted into a routine at each place it is used.

SUBSCRIPT: A number or set of numbers used to specify a particular item in an array.

SWAPPING: In a time-sharing environment, the action of either temporarily bringing a user program into core or storing it on the system device.

SWITCH: A device or programming technique for making selections.

SYMBOL TABLE: A table in which symbols and their corresponding values are recorded.

SYMBOLIC ADDRESS: A set of characters used to specify a memory location within a program.

SYMBOLIC EDITOR: A system library program which helps users in the preparation and modification of source language programs by adding, changing or deleting lines of text.

SYSTEM: A combination of software and hardware which performs specific processing operations.

TABLE: A collection of data stored for ease of reference, generally as an array.

TEMPORARY REGISTER (TEMP): A register which is used primarily as a latch for the result and ALU operation before it is sent to the destination register to avoid race conditions.

TEMPORARY STORAGE: Storage locations reserved for immediate results.

TERMINAL: A peripheral device in a system through which data can enter or leave the computer.

TIMESHARING: A method of allocating central processor time and other computer resources to multiple users so that the computer, in effect, processes a number of programs simultaneously.

TIME QUANTUM: In time-sharing, a unit of time allotted to each user by the monitor.

TOGGLE: To use switches to enter data into the computer memory.

TRANSLATE: To convert from one language to another.

TRUNCATION: The reduction of precision by dropping one or more of the least significant digits, e.g. 3.141592 truncated to four decimal digits is 3.141.

UNDERFLOW: A condition that occurs when a floating point operation yields a result whose magnitude is smaller than the program is capable of expressing.

USER: Programmer or operator of a computer.

VARIABLE: A symbol whose value changes during execution of a program.

WORD: With the IM6100, a 12-bit unit of data which may be stored in one addressable location.

WRITE: To transfer information from memory to a peripheral device or to auxiliary storage.

ZERO PAGE: The first page in the subdivided memory.

ZOMBIE: Appearance assumed by programmer attempting to debug undocumented object code.

APPENDIX F
ASCII CHARACTER CODES

CHARACTER CODES

8-bit ASCII CODE	6-bit CODE	CHARACTER REPRESENTATION	REMARKS
240	40		space (non-printing)
241	41	!	exclamation point
242	42	"	quotation marks
243	43	#	number sign
244	44	\$	dollar sign
245	45	%	percent
246	46	&	ampersand
247	47	'	apostrophe or acute accent
250	50	(opening parenthesis
251	51)	closing parenthesis
252	52	*	asterisk
253	53	+	plus
254	54	,	comma
255	55	-	minus sign or hyphen
256	56	.	period or decimal point
257	57	/	slash
260	60	0	
261	61	1	
262	62	2	
263	63	3	
264	64	4	
265	65	5	
266	66	6	
267	67	7	
270	70	8	
271	71	9	
272	72	:	colon
273	73	;	semicolon
274	74	<	less than
275	75	=	equals
276	76	>	greater than
277	77	?	question mark

<u>8-bit ASCII CODE</u>	<u>6-bit CODE</u>	<u>CHARACTER REPRESENTATION</u>	<u>REMARKS</u>
300	00	@	at sign ¹
301	01	A	
302	02	B	
303	03	C	
304	04	D	
305	05	E	
306	06	F	
307	07	G	
310	10	H	
311	11	I	
312	12	J	
313	13	K	
314	14	L	
315	15	M	
316	16	N	
317	17	O	
320	20	P	
321	21	Q	
322	22	R	
323	23	S	
324	24	T	
325	25	U	
326	26	V	
327	27	W	
330	30	X	
331	31	Y	
332	32	Z	
333	33	[opening bracket, SHIFT/K
334	34	\	backslash, SHIFT/L
335	35]	closing bracket, SHIFT/M
336	36	↑	up arrow
337	37	←	back arrow ²

Footnotes:

(1) In 6-bit code, 00g represents CARRIAGE RETURN

(2) In 6-bit code, 37g represents TAB

CONTROL CODES

<u>8-bit ASCII CODE</u>	<u>CHARACTER NAME</u>	<u>REMARKS</u>
000	null	Ignored in ASCII input
200	leader/trailer	Leader/trailer code precedes and follows the data portion of binary files
203	CTRL/C	(1) IFDOS break character, forces return to Keyboard Monitor, echoed as +C
207	BELL	CTRL/G
211	TAB	CTRL/I, horizontal tabulation
212	LINE FEED	(2) Used as a control character by the Command Decoder and ODT
213	VT	CTRL/K, vertical tabulation
214	FORM	CTRL/L, form feed
215	RETURN	Carriage return, generally echoed as carriage return followed by a line feed
217	CTRL/O	Break Character, used conventionally to suppress Teletype output, echoed as +0
225	CTRL/U	Delete current input line, echoes as +U
232	CTRL/Z	(3) End-of-File character for all ASCII and binary files (in relocatable binary files CTRL/Z is not a terminator if it occurs before the trailer code)
233	ESC	Escape replaces ALTMODE on some terminals Considered equivalent to ALTMODE
375	ALTMODE	Special break character for Teletype input
376	PREFIX	PREFIX replaces ALTMODE on some terminals. Considered equivalent to ALTMODE
377	RUBOUT	Key is labeled DELETE on some terminals Deletes the previous character typed

(1) IFDOS break character--does not affect INTERCEPT JR. MONITOR

(2) OCTAL DEBUGGING TECHNIQUE program as supplied on IM6312 ROM

(3) Applies to IFDOS (INTERSIL FLOPPY DISK OPERATING SYSTEM)

APPENDIX G
LOADING CONSTANTS INTO THE ACCUMULATOR

MNEMONIC	DECIMAL CONSTANT	OCTAL CODE	INSTRUCTIONS COMBINED
K0000 =	0	7300	CLA CLL
K0001 =	1	7301	CLA CLL IAC
K0002 =	2	7305	CLA CLL IAC RAL
		(or)	
K0002 =	2	7326	CLA CLL CML RTL
K0003 =	3	7325	CLA CLL CML IAC RAL
K0004 =	4	7307	CLA CLL IAC RTL
K0006 =	6	7327	CLA CLL CML IAC RTL
K0100 =	64	7203	CLA IAC BSW
K2000 =	1024	7332	CLA CLL CML RTR
K3777 =	2047	7350	CLA CLL CMA RAR
K4000 =	-0	7330	CLA CLL CML RAR
K5777 =	-1025	7352	CLA CLL CMA RTR
K6000 =	-1024	7333	CLA CLL CML IAC RTL
K7775 =	-3	7346	CLA CLL CMA RTL
K7776 =	-2	7344	CLA CLL CMA RAL
K7777 =	-1	7340	CLA CLL CMA

APPENDIX H

OPERATION OF THE PHASELOCK LOOP

INTRODUCTION

The phaselock loop (PLL) is an analog circuit that is available as a single integrated circuit but is in fact a system composed of a few different analog circuits combined together on one chip. The detailed analysis of a PLL uses the mathematics of servomechanism systems. Although there are times that mathematical analysis is desirable, it is not needed to gain a basic understanding of PLL operation. If the following description of PLL operation is studied carefully, the reader will gain the required knowledge to apply the PLL to new designs and to understand how the PLL operates in existing systems. Figure 1 shows the block diagram of a PLL system.

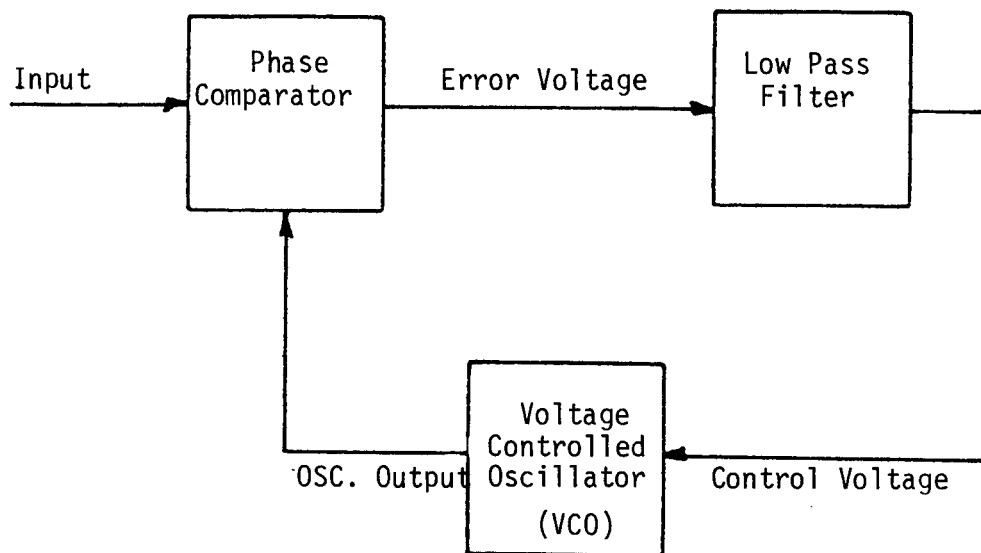


Figure 1

The phase of an input signal and an internal frequency are compared at the phase comparator. Any phase difference between the input signal and the internal frequency produces an error voltage at the output of the phase comparator. The error voltage is filtered by a low-pass filter and is applied as a control voltage to the voltage controlled oscillator (VCO). The VCO is an oscillator whose frequency is controlled by a voltage. The control voltage changes the frequency of the oscillator so as to track the input signal frequency.

A better understanding of PLL operation can be acquired by considering the phase comparator, the low-pass filter, and the VCO as separate elements--each with its own input and output. The individual components can then be analyzed as a closed loop system

PHASE COMPARATOR

The type of phase comparator that is easiest to understand is the type which has a sinewave input and a squarewave VCO internal frequency. Sinewave input and squarewave VCO is a common PLL configuration. When other types of signals are used, the principles are the same but the waveshapes are harder to visualize. Referring to figure 2, this type

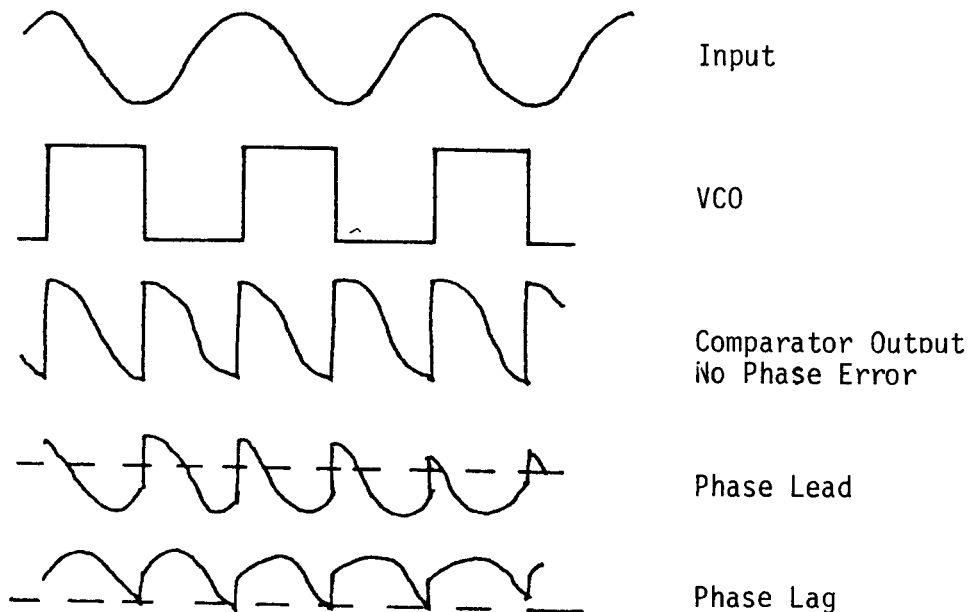


Figure 2

of phase comparator acts as a switchable inverter--on the positive half of the VCO the phase comparator acts as a noninverting amplifier. During the negative half of the VCO cycle, the phase comparator is effectively an inverting amplifier. Note in the waveshapes that there exists a 90° phase shift between the VCO frequency and the input frequency. If the relative phase of the two signals tries to change, the average positive or negative value of the output will shift. A measure of the sensitivity of a comparator is expressed as a gain value equal to:

$$\frac{\text{Change in average output voltage}}{\text{Change in relative phase}} = \frac{dv}{d\theta}$$

LOW PASS FILTER

The low-pass filter takes the phase comparator output and smooths it out to be applied as a d.c. voltage to the VCO. Although many types of filters can be used, the single resistor-capacitor pair is the most frequently used filter. The values used in the filter are seldom critical. If the

low pass filter has too long of a time constant, the PLL will be slow to lock up and will not lock on the desired range of input signals. If the time constant is too short, the PLL may lock on unwanted signals or this VCO will have excessive phase jitter.

VOLTAGE CONTROLLED OSCILLATOR

The VCO is an oscillator whose output frequency is determined by a control voltage. A lower d.c. voltage generates a lower frequency and visa versa. A figure of merit of a VCO is its ability to convert voltage changes into frequency changes. The VCO gain is expressed as:

$$\frac{\text{Change in output frequency}}{\text{Change in control voltage}}$$

THE COMPLETE PLL

Refer to figure 1 to see how the individual components are interconnected to form a closed loop PLL system. As the input signal changes phase or frequency, an error voltage will be generated which changes the VCO frequency so that it is again in lock with the input.

The following terms are frequently used to describe PLL characteristics:

Capture range (f_c)

The range of input frequencies which when applied to the PLL will cause it to lock on to the input signal.

Lock range (f_L)

Once the PLL has locked on to an input signal, the lock range is the frequency band over which the PLL will remain locked. The lock range is always greater than the capture range.

Center frequency (f_0)

The frequency of the VCO when no input signal is applied.

APPLICATIONS OF THE PLL

The PLL can be used as an FM detector. As the frequency of the input varies, the control voltage to the VCO will follow the frequency changes-- therefore the useful output in this case is the control voltage.

The PLL can be used to generate a clean digital signal from a low level input signal. The output signal from the PLL will now be the VCO output.

Frequency multiplication can be accomplished by dividing the VCO frequency before it is applied to the phase comparator, thus the VCO will be an exact multiple of the input frequency as long as the PLL is locked.

There are numerous applications of PLL's and there exist PLL's designed for analog, R.F., and digital applications.

REFERENCES FOR FURTHER STUDY

1. Gardner, F.M., Phaselock Techniques, (Wiley 1966)
2. Viterbi, A.J., Principles of Coherent Communication, (McGraw 1966)
3. National Semiconductor Applications Note AN-46, June 1971
4. Signetics Linear Phase Locked Loops Applications Book, 1972

INTRODUCTION

The remote data station (RDS) board is capable of monitoring a number of different D.C. measurement channels under the control of a microprocessor. The D.C. voltages are converted to a digital format and sent as serial data to the microprocessor. The microprocessor determines which channel is to be measured and when the measurement cycle is to begin. The RDS and microprocessor system communicate via a 4-wire current loop or any other form of 2-way link. When a reading is desired, the microprocessor sends a signal which selects the appropriate channel and simultaneously starts the measurement cycle. After the measurement is complete, the digital value of the D.C. voltage is sent to the microprocessor. The RDS/microprocessor pair is also capable of selecting measurement rate, transmitting and receiving digital control signals, and receiving over/underrange information. Data received by the microprocessor can be used for any of the following operations or any combination of them:

1. Print measured data
2. Perform arithmetic on data
3. Make decisions based on data value
4. Digital control of process functions

Figure 1 is a block diagram of a basic RDS/microprocessor system. A number of D.C. voltages can be sequentially measured by the RDS. Digital input and output signals are also available which are transmitted to and received from the microprocessor.

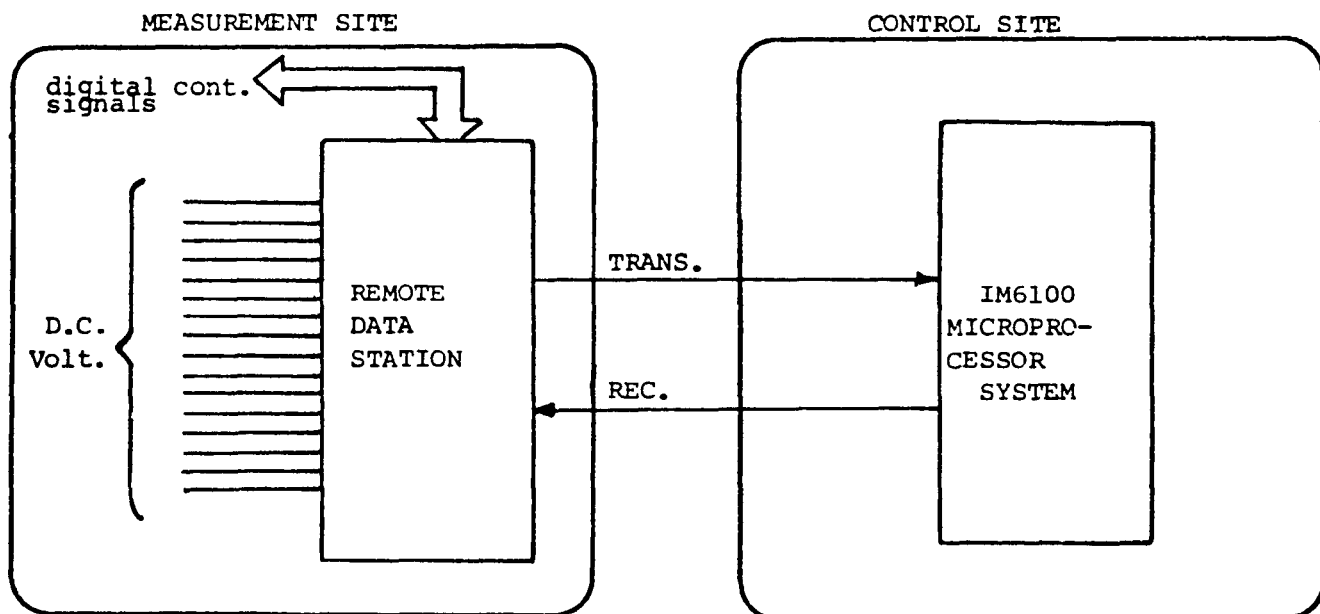


FIGURE 1
BASIC SYSTEM DIAGRAM

Figure 2 is a block diagram of an RDS board. The IH5060 Analog Multiplexer takes one of the D.C. measurement lines and connects it to the input of the A-D converter. The line selected is determined by the control lines coming from the UART. The A-D converter changes the D.C. voltage at its input to binary coded decimal digits. The operation of the IH5060 and 8052A/7103A A-D pair is discussed in the Intersil Analog Products Catalog, Volume I and II. The IM6402 UART takes the converted data and sends it out as serial data to the microprocessor. The receive portion of the UART is used by the microprocessor to determine which channel is selected. An Intersil 7209 oscillator is used as a clock for both the A-D converter and the UART. The 4020 divider provides the necessary frequencies. Transistor circuitry is provided to develop and receive the 20 mA current loops for the communications link. Variations of this link could be RS-232 signals or Modem signals for landline, microwave or RF transmission. The RDS board must be provided with 5 volt \pm 15 volt power connections.

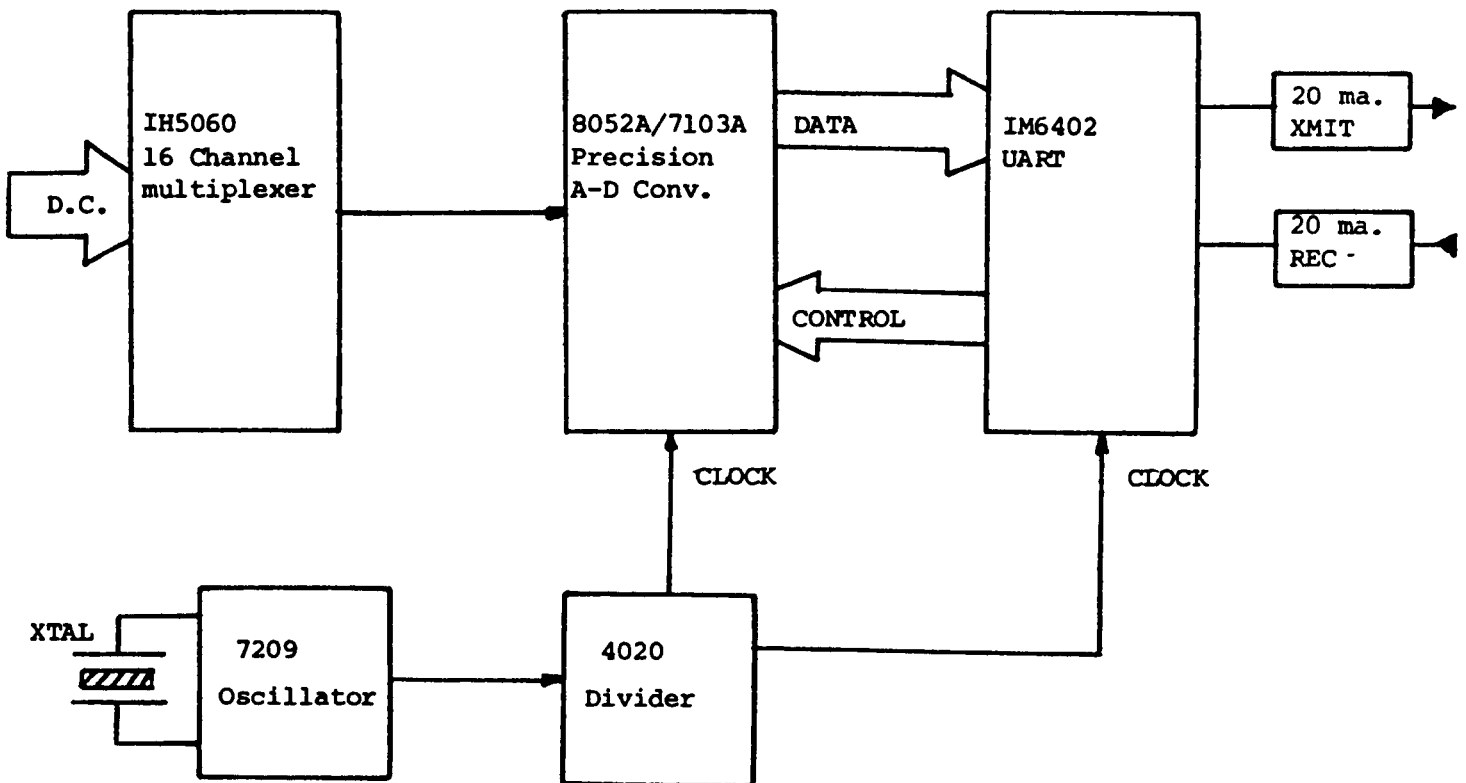


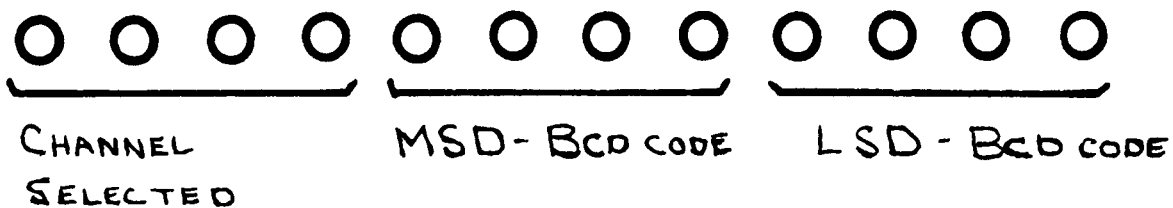
FIGURE 2
RDS BLOCK DIAGRAM

DEMONSTRATION SYSTEM

The demonstration system is a remote data station that measures pressure and transmits the pressure information to the Intercept Jr. microprocessor system.

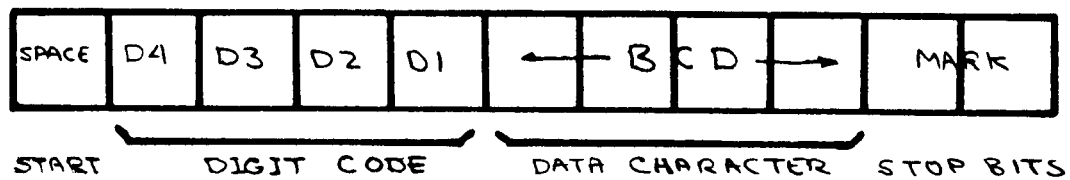
Three optional cards are plugged into the Intercept Jr.; a serial I/O card which is used to communicate with the RDS, a PROM (Programmable Read-Only Memory) card that contains the RDS control program, and a visual display board. The batteries in the Jr. module supply power to both units. The voltage measured from the pressure transducer will light LEDs on the visual display board. The display will represent the transducer offset voltage, ambient pressure voltage, and any pressure voltage generated by squeezing the air bulb. The display format is as follows:

AUDVIS BOARD LEDs



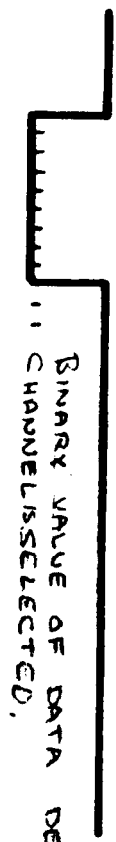
DEMONSTRATION SYSTEM WAVESHAPES

Figure 3 shows the key signals present in the demonstration system. When an RDS reading is requested by the program, a UART TBRL (Transmit Buffer Register Load) signal causes the serial transmit data to be sent to the RDS UART. After the RDS UART receives the serial word, an A-D converter cycle is started. After the A-D cycle is complete, the RDS UART transmits five digits to the microprocessor. Strobe pulses from the A-D converter initiate the transmission of each of the five digits. The data format for each digit is shown below:





MICROPROCESSOR UART "TBRL"



BINARY VALUE OF DATA DETERMINES WHICH CHANNEL IS SELECTED.

UART SERIAL TRANSMIT DATA



RDS UART "RD"



A-D CONVERTER "BUSY"

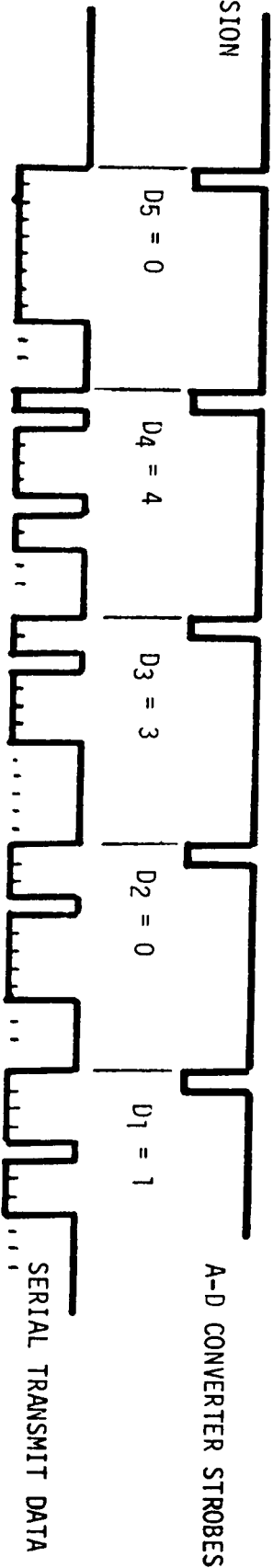


RDS UART "RRD"



A-D CONVERTER INTEGRATIONS

TIME AFTER CONVERSION



A-D CONVERTER STROBES

SERIAL TRANSMIT DATA

FIGURE 3
SYSTEM SIGNALS


```

02000 7201 /          CLA IAC          /DISABLE CONTROL PANEL
02001 6402          6402
02002 4161          CALL          /INITIALIZE PIE CHIP
02003 7445          INPIE

```

/SELECT CHANNEL AND DELAY

```

02004 7200 LOOP:    CLA
02005 4161          CALL          /OUTPUT A CONTROL WORD
02006 7466          TALK
02007 7200          CLA
02010 3100          DCA COUNT
02011 2100          ISZ COUNT      /TIME DELAY LOOP
02012 5211          JMP .-1
02013 7000          NOP

```

/START MEASUREMENT CYCLE

```

02014 7200          CLA
02015 4161          CALL          /OUTPUT A CONTROL WORD
02016 7466          TALK

```

/RECEIVE MEASURED DATA

```

02017 4161          CALL          /READ D5
02020 7501          READ
02021 4161          CALL          /READ D4
02022 7501          READ
02023 4161          CALL          /READ D3
02024 7501          READ
02025 0241          AND MASK
02026 7106          CLL RTL
02027 7106          CLL RTL      /SHIFT BITS LEFT
02030 3101          DCA VALUE      /TEMPORARY STORAGE
02031 4161          CALL          /READ D2
02032 7501          READ
02033 0241          AND MASK      /STRIP OFF VALUE BITS
02034 1101          TAD VALUE      /ADD TO PREVIOUS CHARACTER
02035 6404          DISP          /DISPLAY ON ADVIS BOARD
02036 4161          CALL          /READ D1
02037 7501          READ
02040 5204          JMP LOOP      /CONTINUE LOOPING

```

/CONSTANT:

```

0041 0017 MASK: 0017

```

////////////////////////////////////

CALL 4161
COUNT 0100
DISP 6404
INPIE 7445
LOOP 2004
MASK 2041
READ 7501
TALK 7466
VALUE 0101

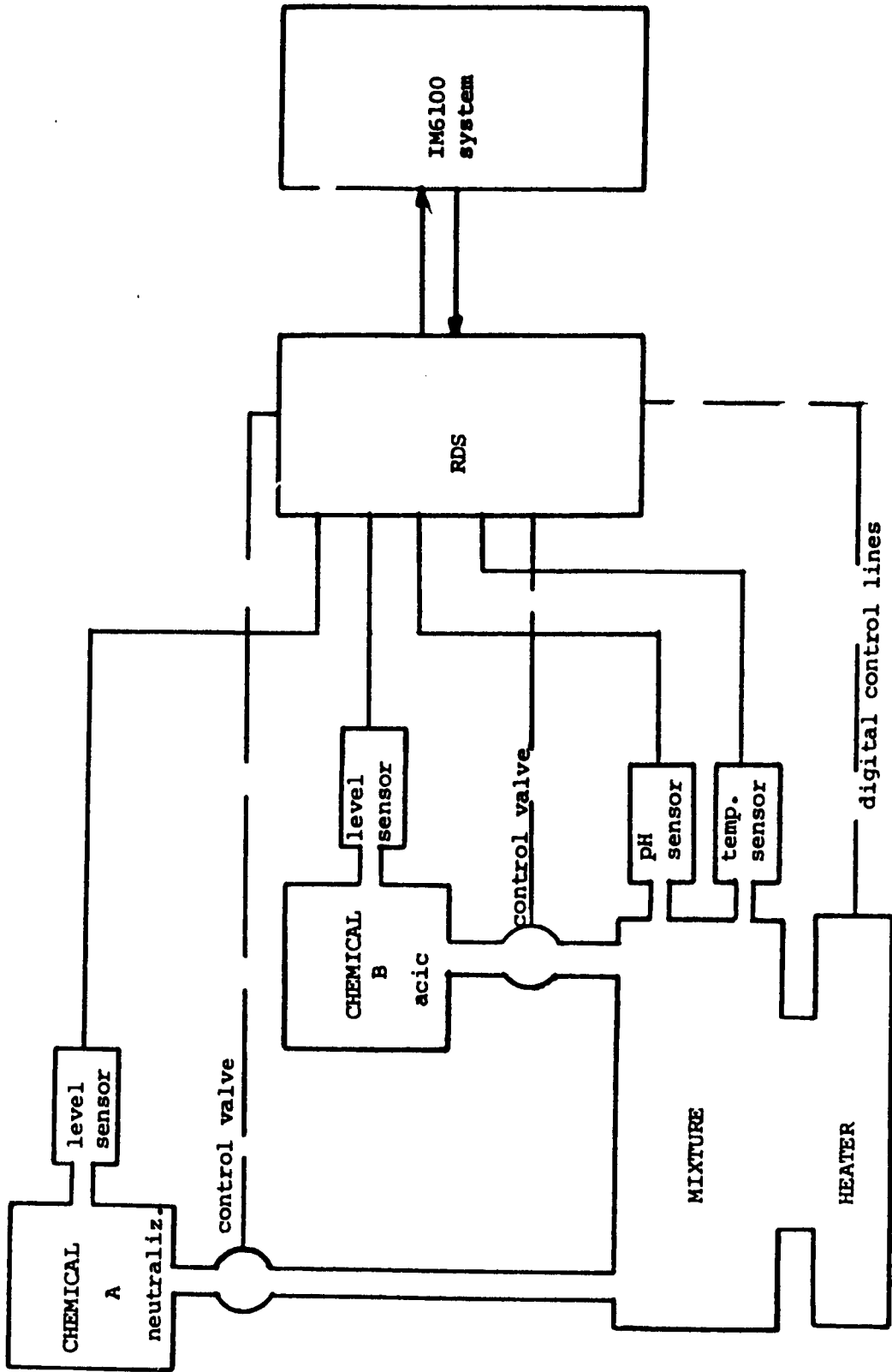
NO ERRORS DETECTED
NO LINKS GENERATED
9 SYMBOLS
5K MEMORY UTILIZED

A FACTORY PROCESS EXAMPLE

In this example (figure 4) we have a factory process in which a mixture is maintained at a constant pH and temperature. The pH is adjusted by adding either chemical A or chemical B to the mixture. An electric heater is used to heat the mixture. A pH sensor converts the mixture pH value to a D.C. voltage and a temperature sensor generates a voltage proportional to the temperature. The levels of both chemical A and B are monitored by sensors which convert the levels to a proportional D.C. voltage. The RDS board accepts all sensor outputs, converts the information to a serial data format and transmits the data to the microprocessor system. The microprocessor continuously monitors the value of pH, if the pH goes beyond an allowable range, the microprocessor sends control signals to the RDS which activate valves for either chemical A or chemical B. Similarly, when the temperature exceeds the preset bounds, the microprocessor sends control signals which either increase or decrease the amount of power applied to the heater element. The microprocessor has the ability to correct certain sensor errors, do self-calibration routines, and to run troubleshooting programs which aid maintenance personnel in correcting production problems. The microprocessor system can either be located next to the RDS or placed some distance away. This flexibility allows the process system designer to place the RDS at the location most suitable to the manufacturing process.

METHODS OF COMMUNICATING WITH THE MICROPROCESSOR

Figure 5 shows the method used on the demonstration unit to communicate with the microprocessor system. A logic one is represented by a 20 mA current (mark), and a logic zero is represented by the absence of a current (space). Figure 12 shows a voltage level link using RS-232 standard signals. For transmission over a significant distance, it is possible to convert the digital signals to sinewave frequencies and transmit the data over landline or radio links. Figure 13 shows a MODEM link. The digital signal is fed to the MODEM (Modulator-Demodulator) which converts a logic one to an audio frequency and a logic zero to a lower audio frequency. The MODEM then sends the audio signal out for transmission over telephone lines or some form of RF link. The RF link could either be a microwave, high-frequency radio, or any other form of radio transmission. At the receive end, the MODEM takes the received audio and converts it back to digital ones and zeros and sends the data to the microprocessor system.



_____ = D.C. measurement lines
 - - - - - = Digital control lines

FIGURE 4
 A FACTORY PROCESS

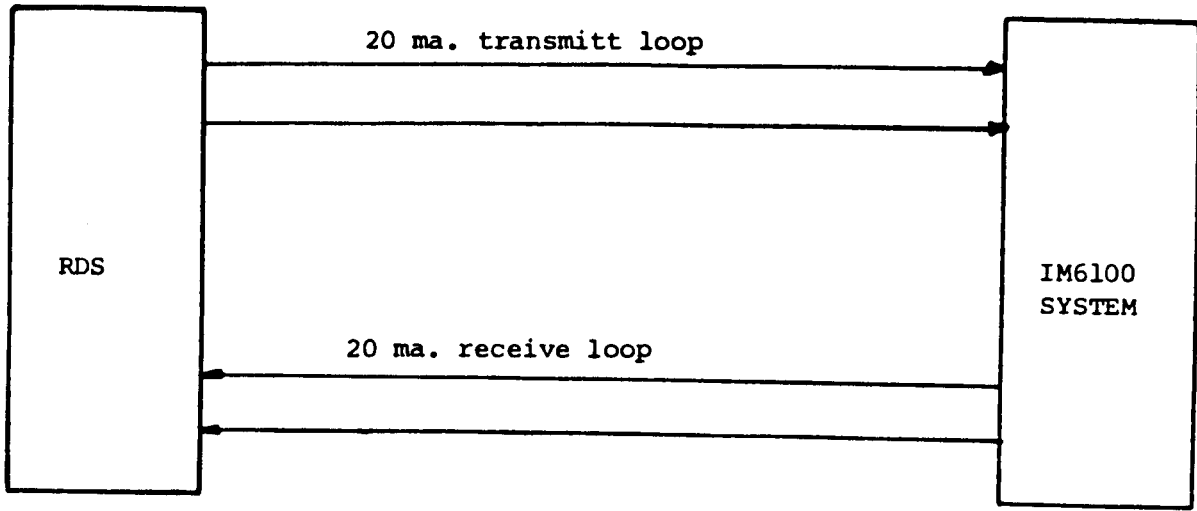


FIGURE 5
CURRENT LOOP

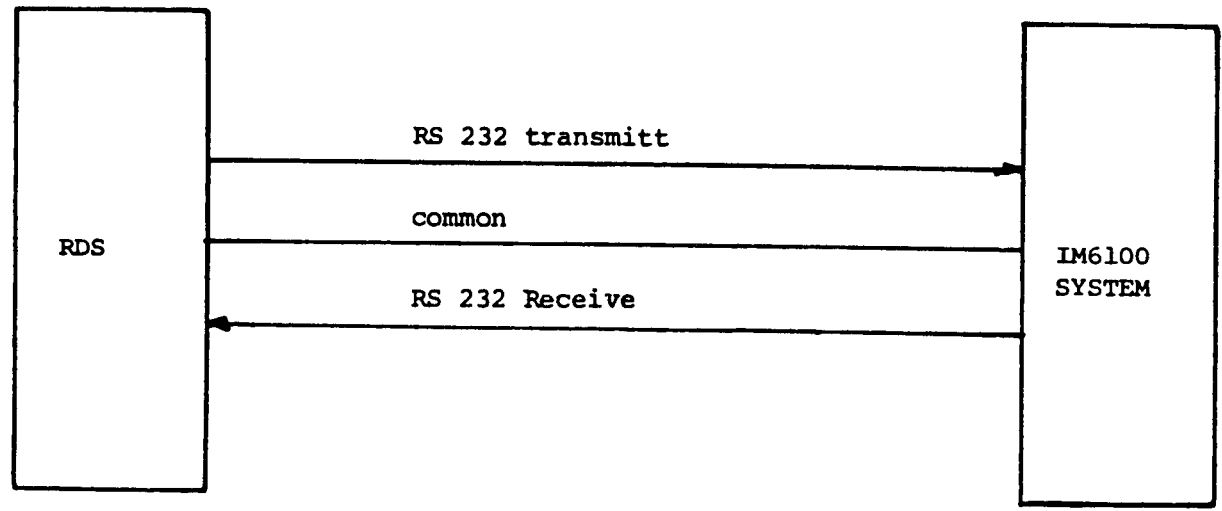


FIGURE 6
RS-232

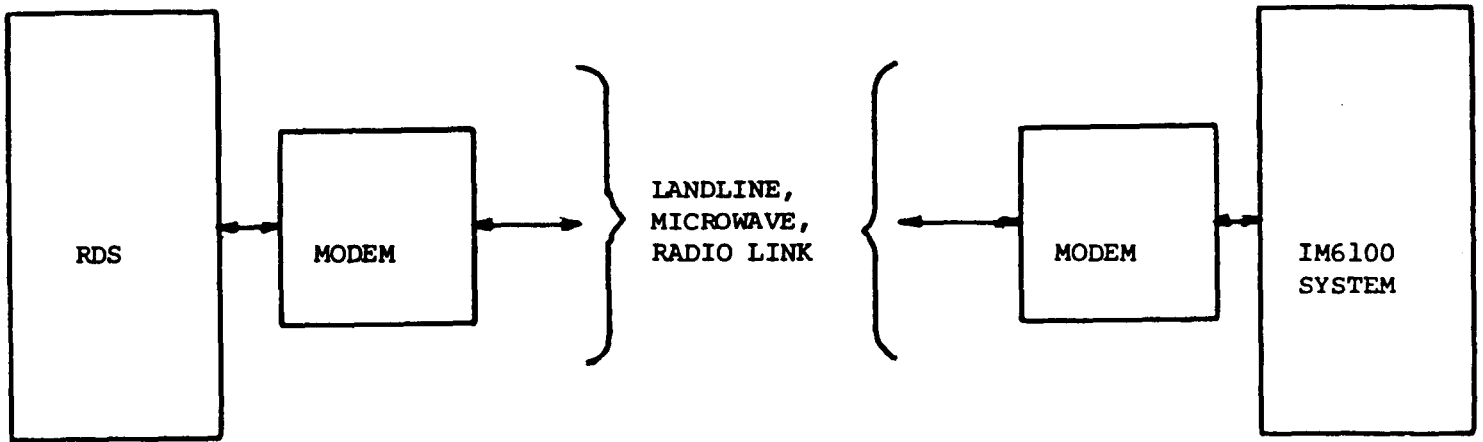


FIGURE 7
MODEM LINK

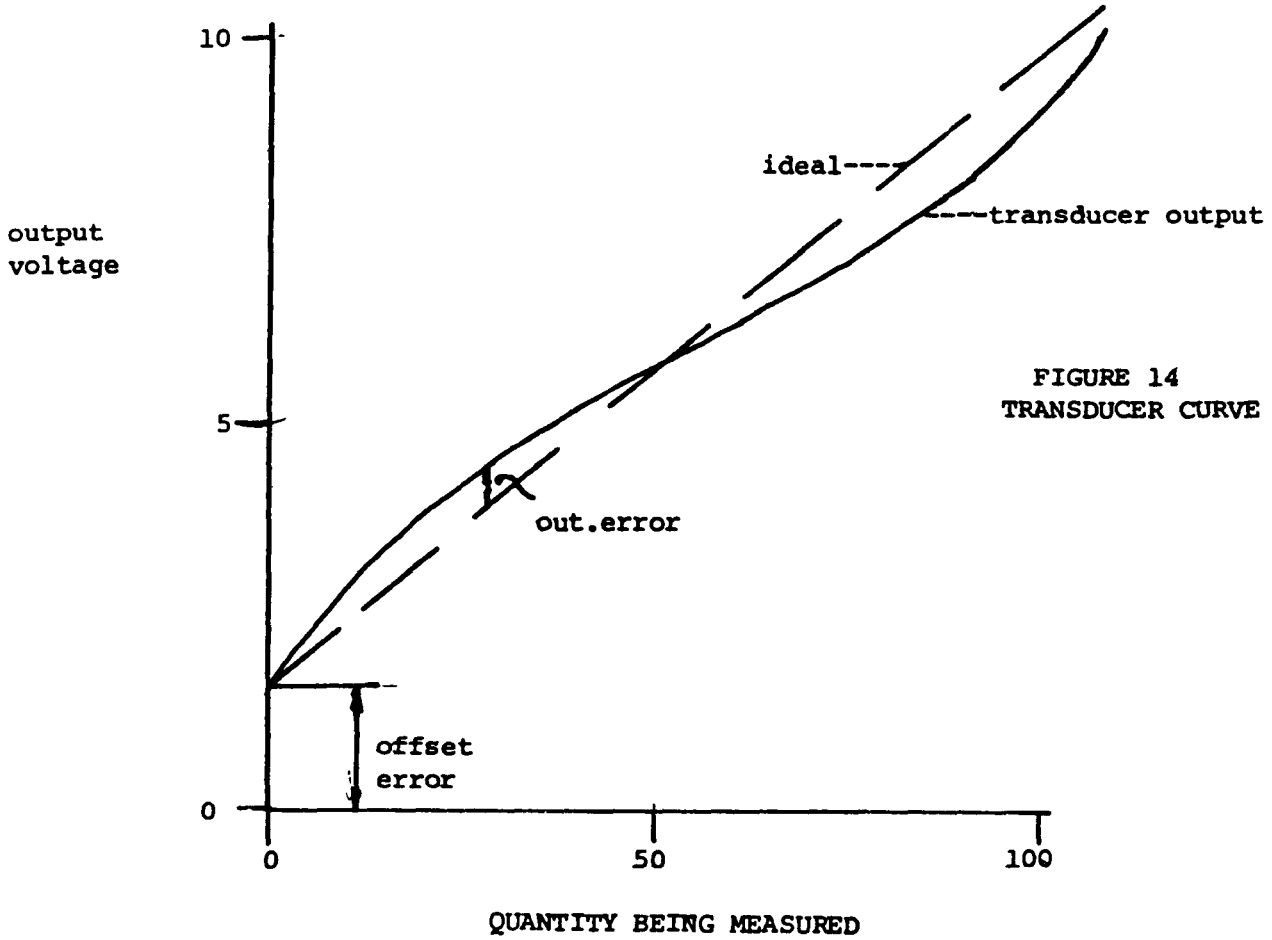


FIGURE 14
TRANSDUCER CURVE

IMPROVING TRANSDUCER CHARACTERISTICS

The RDS/microprocessor pair is capable of measuring transducer signals and applying correction factors to reduce sources of transducer errors. Offset, nonlinearity, and temperature drift are three of the more prominent transducer errors. Figure 14 is a generalized transfer characteristic of a transducer. An offset error is the amount of output voltage present when the input quantity is at its zero reference. Without a microprocessor it is necessary to use active circuits to remove this offset. The microprocessor system is capable of storing the offset value and subtracting it from all measurements. This procedure of storing error values and correcting received data can also be used to correct nonlinearity errors. Depending on the accuracy required and memory available, errors at various points along the transfer characteristic are stored. During a data measurement, the error for that particular reading is referenced and the error is subtracted out. If power or space requirements restrict the ability to hold the transducer at a constant temperature, the microprocessor can make a temperature reading and correct the transducer as required.

THEORY OF DEMONSTRATION SYSTEM

Reference should be made to figure 9 for the following description. The demonstration board is a simplified version of a complete 16-channel data acquisition system. This board is capable of measuring up to 4 D.C. channels.

IC1 is a sixteen channel analog multiplexer. The D.C. inputs are present on pins 19-20. The channel to be selected is determined by the digital inputs on pins 15-17. The output of the multiplexer is available at pin 28 and is sent through R8 to the input of the precision A-D converter pair consisting of IC2 and IC3. The output of the A-D converter is in the form of multiplexed BCD characters. The BCD value is available at IC2 pins 20 to 23 and the digit selected is determined by the signals at pins 24 to 27. Both the BCD information and the digit select information is sent to the UART (IC4) to be transmitted as serial data. When the UART receives serial data from the microprocessor, an analog channel is selected and the A-D converter is commanded to start a measurement cycle. At the end of the measurement cycle the data on T1 through T8 (IC4 pins 26 to 33) is transmitted to the microprocessor. IC6 is an Intersil 7209 oscillator. The 4020 is a divider which provides the clock for both the A-D converter and for the UART. The 20 mA loop current is generated by a 2N3638 and associated components. The 20 mA received signal is converted to standard CMOS levels by the 4069, pins 12 and 13. A complete description of the Intersil parts is available in the Analog Products Catalog and the IM6100 Microprocessor Booklet.

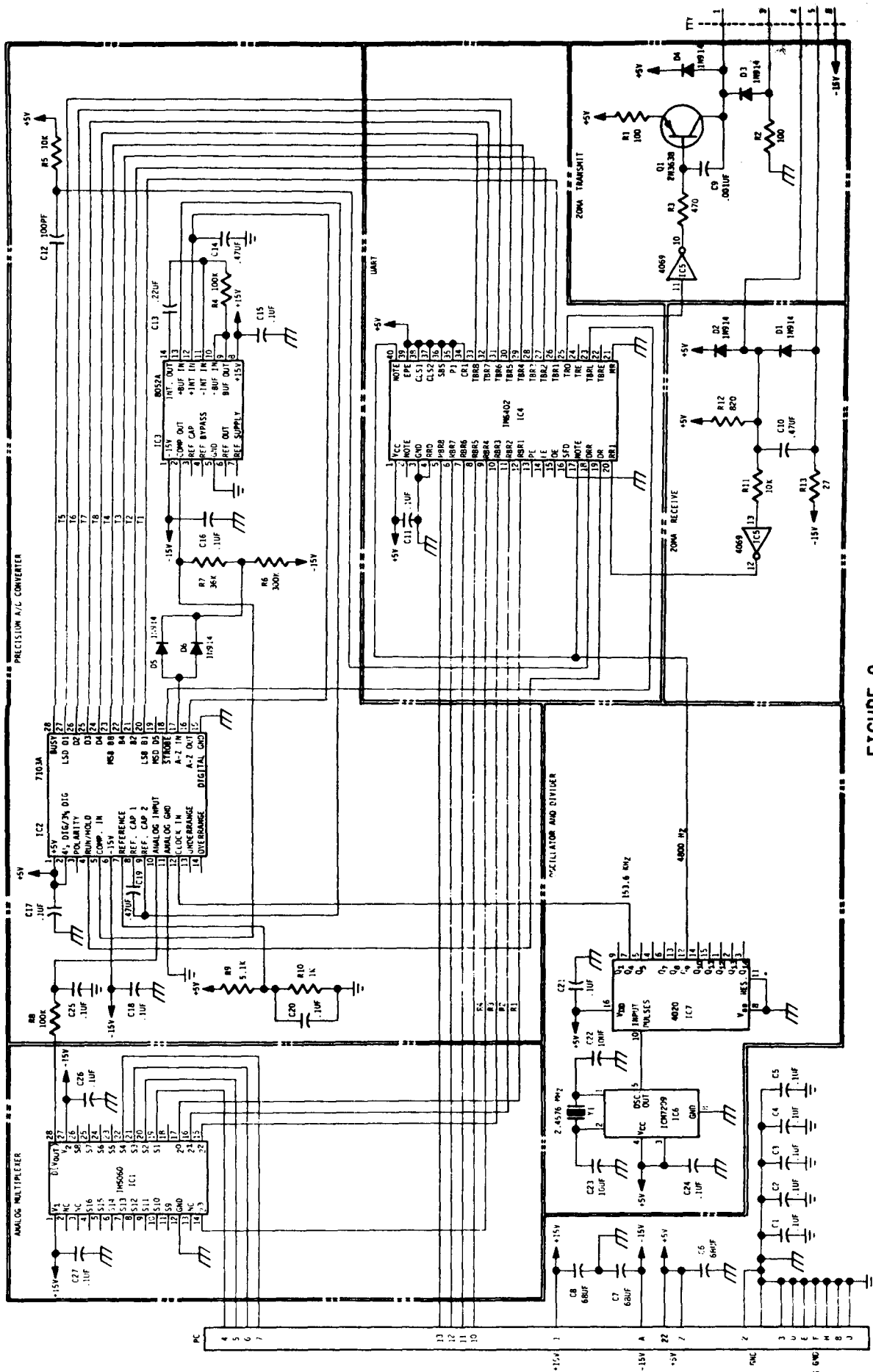


FIGURE 9
RDS SCHEMATIC

APPENDIX J
KEY BOARD TENNIS PROGRAM WITH INTERCEPT JR.

DEMO PROGRAM: "PING"

IN 'PING', THE PLAYER PLAYS AGAINST THE MACHINE. THE COMPUTER "SERVES" FROM THE LEFT, AND THE "BALL" TRAVELS ALONG THE LED'S UNTIL IT REACHES BIT 11, THE RIGHTMOST LED.

IF THE PLAYER PRESSES THE YELLOW BUTTON (IAC), THE BALL WILL BE RETURNED WITH A 'CLICK'. THE MACHINE WILL RETURN THE BALL AND THE SEQUENCE IS REPEATED.

IN ORDER TO ADD EXCITEMENT TO THE GAME, EACH TIME THE PLAYER RETURNS THE BALL, IT SPEEDS UP.

WHEN THE PLAYER MISSES, BY PRESSING THE BUTTON TOO SOON OR TOO LATE, THE MACHINE BUZZES, DELAYS, THEN SERVES AT THE SLOWEST RATE.

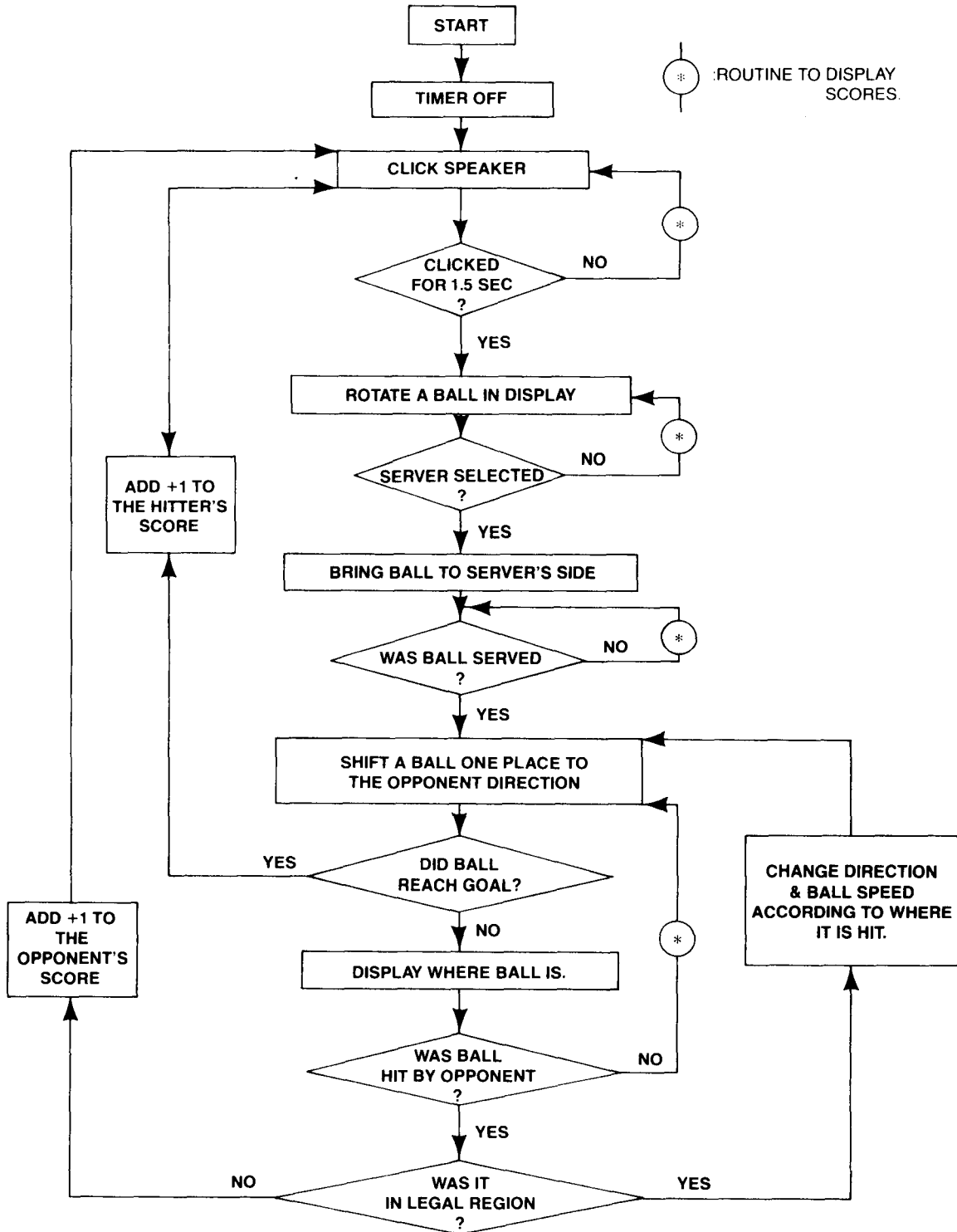
HAVE FUN!

(NOTE: THE CONTENTS OF LOCATION 0262 DETERMINE THE ORIGINAL SPEED OF THE BALL, AND LOCATION 0263 DETERMINES HOW FAST IT SPEEDS UP.)

“PING”

ADDRESS,	CONTENTS,	ADDRESS,	CONTENTS,	ADDRESS,	CONTENTS,
0201	7300	0223	7320	0245	1263
0202	1262	0224	6404	0246	3264
0203	3264	0225	6401	0247	7004
0204	7330	0226	2265	0250	3265
0205	6401	0227	5223	0251	1264
0206	6404	0230	7010	0252	3266
0207	3265	0231	2265	0253	1265
0210	1264	0232	5231	0254	6404
0211	3266	0233	7440	0255	7450
0212	7604	0234	5230	0256	5204
0213	7440	0235	5201	0257	2266
0214	5236	0236	6401	0260	5255
0215	2266	0237	7300	0261	5247
0216	5212	0240	1265	0262	0000
0217	1265	0241	7010	0263	1000
0220	7010	0242	7440	0264	—
0221	7440	0243	5223	0265	—
0222	5206	0244	1264	0266	—

FLOWCHART FOR KEYBOARD TENNIS PROGRAM WITH INTERCEPT JR.



/KEY BOARD TENNIS WITH INTERCEPT JR.

/RULES:

START AT LOCATION #200.
 SINCE JR IS WAITING FOR SIGN OF STARTER,
 PRESS IAC OR CTR WHCIHEVER STARTS FIRST
 TO PREPARE FOR SERVICE.
 THEN, SERVE THE BALL BY PRESSING THE KEY.
 THE OPPONENT MUST PRESS KEY BEFORE BALL
 HITTING THE SIDE BUT IN THE NEAREST 2 BITS
 NOT TO LOSE POINTS.
 SCORE IS +1 FOR ONE SUCCESSFUL GOAL AND
 +1 BY THE OPPONENT'S FAULT. THE HIGHEST
 SCORE WHICH CAN BE HANDLED IS 99.

/DEFINITIONS:

6400 WRITED=6400 /WRITE DISPLAY.
 6401 CLICK=6401 /CLICK SPEAKER.
 6402 TIMER=6402 /TIMER ON OR OFF.
 6404 WRITES=6404 /WRITE DISPLAY OF I/O BRD.

/SUBPROGRAMS:

*20
 00020 0000 KEY, 0 /TO DETECT KEY BOARD.
 00021 7300 CLA CLL
 00022 7604 LAS /LOAD AC WITH SR.
 00023 7004 RAL /CTR=4000.
 00024 7430 SZL /CTR KEY PRESSED?
 00025 5040 JMP ID0 /YES.
 00026 7004 RAL /IAC=2000.
 00027 7620 SNL CLA /IAC KEY PRESSED?
 00030 5420 JMP I KEY /NEITHER PRESSED.
 00031 7140 CLL CMA /PUT ALL 1'S IN AC.
 00032 3120 DCA ID /ID=1 FOR IAC PLAYER.
 00033 7604 GO, LAS /STOP FURTHER EXECUTION
 00034 7640 SZA CLA /UNTIL KEY IS RELEASED.
 00035 5033 JMP .-2
 00036 2020 ISZ KEY /TO GET OUT OF WAITING LOOP.
 00037 5420 JMP I KEY
 00040 7300 ID0, CLL CLA
 00041 3120 DCA ID /ID=0 FOR CTR PLAYER.
 00042 5033 JMP GO /RETURN.

/SUBROUTINE TO DISPLAY SCORES:
 00043 0000 SHOW, 0
 00044 7300 CLA CLL
 00045 3117 DCA DIGIT2 /CLEAR REGISTER DIGIT2.
 00046 1124 TAD SCORE1 /BRING IAC PLAYER'S SCORE.
 00047 4074 JMS DECIML /CONVERT OCTAL TO DECIMAL.
 00050 1116 TAD DIGIT1 /1ST DECIMAL DIGIT.
 00051 3121 DCA SAVE1 /STORE IT IN SAVE1.
 00052 1117 TAD DIGIT2 /STORE 2ND DECIMAL DIGIT
 00053 3122 DCA SAVE10 /IN SAVE10.
 00054 3117 DCA DIGIT2 /CLEAR DIGIT2.
 00055 1125 TAD SCORE2 /BRING CTR PLAYER'S SCORE.
 00056 4074 JMS DECIML /CONVERT IT INTO DECIMAL NO.
 00057 1116 TAD DIGIT1 /SHIFT 1ST DECIMAL NO. INTO
 00060 7106 CLL RTL /2ND BITE FROM RIGHT.
 00061 7006 RTL
 00062 1121 TAD SAVE1 /JOIN TO IAC PLAYER'S SCORE.
 00063 1133 TAD K4000 /SET BIT #0 TO DISPLAY THEM.
 00064 4106 JMS DELAY /DISPLAY 1ST DECIMAL DIGITS
 00065 1117 TAD DIGIT2 /OF BOTH PLAYER'S SCORES.
 00066 7106 CLL RTL /SHIFT 2ND DECIMAL NO.
 00067 7006 RTL
 00070 1122 TAD SAVE10 /JOIN TO IAC PLAYER'S SCORE.
 00071 1132 TAD K2000 /SET BIT #1.
 00072 4106 JMS DELAY /DISPLAY 2ND DECIMAL DIGITS
 00073 5443 JMP I SHOW /OF BOTH SCORES & RETURN.

/SUBROUTINE TO CONVERT OCTAL TO DECIMAL.
 00074 0000 DECIML, 0
 00075 7100 CLL /KILL LINK BIT.
 00076 1127 TAD M12 /AC=-12.
 00077 7420 SNL /NO MORE 2ND DECIMAL DIGITS?
 00100 5103 JMP OUT /IF NOT, OUTPUT RESULTS.

```

00101 2117      ISZ DIGIT2      /IF YES, COUNT THE DIGITS.
00102 5075      JMP DECIML+1    /EXHAUST 10TH DIGIT.
00103 1130      OUT,      TAD P12          /ADD 10 TO COMPENSATE.
00104 3116      DCA DIGIT1    /STORE IT.
00105 5474      JMP I DECIML

/
/
00106 0000      DELAY, 0      /SUBROUTINE TO DISPLAY & DELAY TIME.
00107 6400      WRITES      /DISPLAY AC CONTENTS.
00110 7300      CLA CLL
00111 1131      TAD M7000    /TO COUNT 512.
00112 3123      DCA TEMP
00113 2123      ISZ TEMP      /COMPLETED COUNTING?
00114 5113      JMP -1         /NOT YET.
00115 5506      JMP I DELAY     /YES.

/
/
/ DATA (1):
/
00116 0000      DIGIT1, 0
00117 0000      DIGIT2, 0
00120 0000      ID, 0
00121 0000      SAVE1, 0
00122 0000      SAVE10, 0
00123 0000      TEMP, 0
00124 0000      SCORE1, 0
00125 0000      SCORE2, 0
00126 7774      M4, -4
00127 7766      M12, -12
00130 0012      P12, 0012
00131 7000      M7000, 7000
00132 2000      K2000, 2000
00133 4000      K4000, 4000
00134 7700      K7700, 7700

/PROGRAM FOR GAME STARTS HERE:
*200 /STARTING ADDRESS.
00200 7301      CLA CLL IAC /AC=1 FOR TIMER OFF.
00201 6402      TIMER      /AC MUST BE 0 FOR TIMER ON.
00202 7200      CLA
00203 3124      DCA SCORE1 /INITIAL SCORE.
00204 3125      DCA SCORE2
00205 1134      DISPLY, TAD K7700 /CLICK SPEAKER 64 TIMES
00206 3361      DCA COUNT   /IN 1.5 SEC FOR STARTING SIGN.
00207 6401      CLICK
00210 4043      JMS SHOW    /TO KEEP DISPLAYING.
00211 2361      ISZ COUNT
00212 5207      JMP -3
00213 7301      CLA CLL IAC /AC=1.
00214 6404      RLEFT, WRITES /DISPLAY AC.
00215 3362      DCA SR      /SAVE DISPLAY BIT.
00216 4355      JMS BOARD  /CHECK KEY COMMAND.
00217 4043      JMS SHOW    /40 MS TIME DELAY
00220 4043      JMS SHOW    /TO KEEP DISPLAYING.
00221 1362      TAD SR      /BRING DISPLAY BIT BACK.
00222 7004      RAL        /SHIFT LEFT ONE.
00223 7420      SNL        /REACHED TO EDGE?
00224 5214      JMP RLEFT   /NOT YET.
00225 7010      RAR        /YES.
00226 6404      RRIGHT, WRITES /DISPLAY.
00227 3362      DCA SR      /SAVE IT.
00230 4355      JMS BOARD  /CHECK KEY INPUT.
00231 4043      JMS SHOW    /40 MS TIME DELAY TO
00232 4043      JMS SHOW    /DISPLAY.
00233 1362      TAD SR
00234 7010      RAR        /SHIFT RIGHT ONE.
00235 7420      SNL        /REACHED TO EDGE?
00236 5226      JMP RRIGHT  /IF NOT, KEEP SHIFTING.
00237 7004      RAL        /IF YES, CHANGE DIRECTION.
00240 5214      JMP RLEFT
00241 1120      START, TAD ID   /CHCK WHICH PLAYER FIRST.
00242 7650      SNA CLA    /THE FOLLOWING ROUTINE
00243 5251      JMP +6     /BRINGS BALL TO THE
00244 7101      CLL IAC    /PLAYER'S SIDE.
00245 6404      WRITES
00246 3362      DCA SR      /SAVE DISPLAY BIT.
00247 1364      TAD LEFT   /LEFT=RAL.
00250 5255      JMP +5
00251 1133      TAD K4000
00252 6404      WRITES
00253 3362      DCA SR      /SAVE DISPLAY BIT.
00254 1365      TAD RIGHT  /RIGHT=RAR.
00255 3266      DCA ROTATE /DEFINE SHIFT DIRECTION.
00256 4020      JMS KEY    /GAME STARTED?
00257 5261      JMP +2     /NOT YET.
00260 5263      JMP +3     /YES, STARTED.
00261 4043      JMS SHOW    /TO KEEP DISPLAYING.
00262 5256      JMP -4     /CHECK KEY AGAIN.
00263 1274      TAD SPEED+1 /INITIALIZE SPEED.
00264 3273      DCA SPEED

```

```

00265 1362 TAD SR /BRING DISPLAY BIT TO SHIFT.
00266 0000 ROTATE, 0 /RAL OR RAR IS STORED HERE.
00267 7430 SZL /SUCCEEDED TO GOAL?
00270 5341 JMP SCORE /IF YES, SCORE & CLICK.
00271 6404 WRITES /DISPLAY NEW SHIFTED BIT.
00272 3362 DCA SR /SAVE DISPLAY BIT.
00273 4043 SPEED, JMS SHOW /THIS IS ONLY FOR SERVER.
00274 4043 JMS SHOW /20 MS TIME DELAY.
00275 4043 JMS SHOW /20 MS.
00276 4043 JMS SHOW /20 MS.
00277 4043 JMS SHOW /FASTEST=40 MS, SLOWEST=100 MS.
00300 4020 JMS KEY /OPPONENT KEY PRESSED?
00301 5265 JMP ROTATE-1 /NOT YET, SO KEEP SHIFTING.
00302 1120 TAD ID
00303 7640 SZA CLA /WHICH PLAYER RECEIVED BALL?
00304 5321 JMP CTR /CTR SIDE.
00305 1362 TAD SR /IAC SIDE.
00306 1131 TAD M7000 /DETERMINE RETURN SPEED.
00307 7440 SZA /HIT BALL AT 2ND BIT?
00310 5314 JMP AI /NO.
00311 1131 EASY, TAD M7000 /IF YES, GIVE EASY BALL.
00312 3273 DCA SPEED /M7000="NOP".
00313 5330 JMP CHANGE /RETURN THE BALL.
00314 7710 AI, SPA CLA /IS IT FAULT OR BEST BIT?
00315 5351 JMP FAULT /HIT IN WRONG REGION.
00316 1363 DFFCLT, TAD JMPDFF /IT WAS BEST HIT. SO, RETURN
00317 3273 DCA SPEED /BALL FASTEST.
00320 5330 JMP CHANGE
00321 1362 CTR, TAD SR
00322 1126 TAD M4 /AC=-4.
00323 7450 SNA /HIT AT 2ND BIT.
00324 5311 JMP EASY /YES.
00325 7700 SMA CLA /IS IT FAULT HIT?
00326 5351 JMP FAULT /YES.
00327 5316 JMP DFFCLT /NO, IT WAS BEST HIT.
00330 1120 CHANGE, TAD ID /CHANGE DIRECTION.
00331 7650 SNA CLA
00332 5335 JMP .+3
00333 1364 TAD LEFT
00334 5336 JMP .+2
00335 1365 TAD RIGHT
00336 3266 DCA ROTATE /DEFINE NEW DIRECTION.
00337 7100 CLL /CLEAR USELESS LINK BIT.
00340 5265 JMP ROTATE-1 /SHIFT TO THE DIRECTION.
00341 7300 SCORE, CLA CLL
00342 1120 TAD ID
00343 7650 SNA CLA /WHICH SCORED?
00344 5347 JMP .+3
00345 2124 ISZ SCORE1 /IAC SIDE.
00346 5205 JMP DISPLY
00347 2125 ISZ SCORE2 /CTR SIDE.
00350 5205 JMP DISPLY
00351 1120 FAULT, TAD ID /CHECK WHO WAS AGAINST RULE.
00352 7040 CMA /GIVE POINT TO THE OPPONENT.
00353 3120 DCA ID
00354 5342 JMP SCORE+1
/
00355 0000 BOARD, 0 /SUBROUTINE BOARD.
00356 4020 JMS KEY /CHECK KEY.
00357 5755 JMP I BOARD /IF NO INPUT, RETURN TO LOOP.
00360 5241 JMP START /IF SIGN, START GAME.
/
/
/ DATA (2):
/
00361 0000 COUNT, 0
00362 0000 SR, 0
00363 5276 JMPDFF, 5276
00364 7004 LEFT, 7004
00365 7010 RIGHT, 7010
/
/
00366 7000 NOP

```

A1 0314
BOARD 0355
CHANGE 0330
CLICK 6401
COUNT 0361
CTR 0321
DECIML 0074
DELAY 0106
DFFCLT 0316
DIGIT1 0116
DIGIT2 0117
DISPLY 0205
EASY 0311
FAULT 0351
GO 0033
ID 0120
ID0 0040
JMPDFF 0363
KEY 0020
K2000 0132
K4000 0133
K7700 0134
LEFT 0364
M12 0127
M4 0126
M7000 0131
OUT 0103
P12 0130
RIGHT 0365
RLEFT 0214
ROTATE 0266
RRIGHT 0226
SAVE1 0121
SAVE10 0122
SCORE 0341
SCORE1 0124
SCORE2 0125
SHOW 0043
SPEED 0273
SR 0362
START 0241
TEMP 0123
TIMER 6402
WRITED 6400
WRITES 6404

APPENDIX K

OCTAL DEBUGGING TECHNIQUE ROM

IM6312-001 CMOS OCTAL DEBUGGING TECHNIQUE (ODT) ROM

The 63S003 version of the INTERSIL ODT ROM may be used in the INTERCEPT JR. in place of the IM6312-002 MONITOR ROM. With the addition of a 6953-PIEART serial I/O board and a 110 baud ASCII terminal, the ODT program provides the user an easy way to enter, edit and execute programs in the INTERCEPT JR. from a terminal.

Upon operating the RESET switch, the following sequence is executed:

7772	7301	CLA CLL IAC	/SET AC TO 0001
7773	6400	6400	/CLEAR INTERCEPT JR. DISPLAY
7774	6402	6402	/TURN OFF CP TIMER
7775	5776	JMP.+1	/GO TO START OF ODT
7776	6000	INIT	/START ADDRESS OF ODT
7777	5372	JMP.-5	/ENTRY POINT FOR CP REQUEST

This results in a blanked display, disabled CP timer and ODT running in control panel memory. ODT may be run in main memory (thus allowing all instructions to work normally) by entering the following instructions through the terminal and executing them:

0200	6407	IOT RUN	/RESET CP FF AFTER
0201	6001	ION	/EXECUTING NEXT INSTRUCTION
0202	5603	JMP.+1	/GO TO JR
0203	7777	7777	/INITIALIZATION SEQUENCE

INTERCIL ODT requires a RUN/HLT switch and manipulation of punch ON/OFF switch for proper tape punch operation so it is recommended that the MONITOR ROM memory dump routines be used to punch tape. INTERCIL ODT also contains commands for working with multiple fields. These commands have no effect in the INTERCEPT JR. which does not contain hardware for handling fields. All other ODT functions will run properly with INTERCEPT JR. hardware.

The following is a summary of the keyboard commands used with ODT:

nnnn/ Open location designated by octal number nnnn. When a location is opened, its content is printed and may be altered.

 Reopen latest opened location.

CR	Closes location
LF	Closes location, then opens next sequential location
(SHIFT/O)	Closes location and opens indirect reference (content of location taken as an address, and new location is opened)
(SHIFT/N)	Closes location and opens new location referenced when contents of old location are treated as a memory reference instruction
nnnnG	Begin executing program at nnnn
nnnnB	Set breakpoint at nnnn
C	Resume execution (continue after breakpoint)
A	Examine/modify AC, MQ, L
M	Open search mask, lower bound upper bound
nnnnW	Search memory between specified bounds for octal value nnnn
T	Punch header/trailer
nnnn;mmmmP	Punch binary memory image defined by limits nnnn and mmmm
E	Punch checksum and trailer
L	Load BIN tape from tape reader and print checksum at end of load

For complete documentation, consult the INTERSIL CMOS Family Sampler Manual and Change Notice SAMPLR 001 in Appendix L.

ODT LISTING

```

57.      BEGIN PASS 2
58.
59.      1      /ODT-F VERSION 5 TAPE 1
60.      2      /ODT-F
61.      3      /OCTAL DEBUGGING TECHNIQUE PROGRAM
62.      4      /WITH CAPACITY FOR HANDLING FIELDS
63.      5      /AND BROKEN INTO ROM AND RAM SECTIONS
64.      6
65.      7      / COMMANDS ARE THE SAME AS FOR DEC ODT PLUS
66.      8
67.      9      / N. -- SET CURRENT FIELD TO FIELD N
68.     10      / (CURRENT FIELD IS SAME AS INSTRUCTION FIELD)
69.     11      / A<LF><LF> -- OPENS A REGISTER EQUIVALENT TO M0
70.     12      / A<LF><LF><LF> -- OPENS A REGISTER CONTAINING DATA FIELD
71.     13      / THESE COMMANDS HAVE NO EFFECT OR PRODUCE HARMLESS GARBAGE
72.     14      / ON IM6100 SYSTEMS AND OTHER SYSTEMS WHICH DO NOT HAVE
73.     15      / EMA CAPABILITY.
74.     16
75.     17
76.     18      / L. -- LOAD FROM THE TAPE READER USING BIN FORMAT
77.     19      / (WILL IGNORE CHANGE FIELD CHARACTERS)
78.     20      / TO USE COMMAND, GIVE AN L AFTER THE PROMPT THEN PLACE TAPE
79.     21      / IN READER ON LEADER-TRAILER AND START THE TAPE READER.
80.     22      / THE BIN TAPE WILL BE READ INTO THE CURRENT FIELD AND
81.     23      / THE CHECKSUM WILL BE PRINTED OUT ON THE TTY FOLLOWING
82.     24      / THE END OF THE LOAD.
83.     25
84.     26
85.     27
86.     28
87.     29
88.     30
89.     31      / ***** DEFINITIONS *****
90.     32      0000 F1=0      /FIELD OCT-F IS IN
91.     33      0000 F1A=00    /FIELD OCT-F IS IN, IN BITS 6-8
92.     34      0005 ZPAT=5    /FIRST ADDRESS FOR TWO WORDS OF BREAKPOINT
93.     35      /LINKAGE IN FIELD OF PROGRAM TO BE DEBUGGED
94.     36      6000 START=6000 /STARTING LOCATION FOR ODT-F
95.     37      / THE I/O INSTRUCTIONS
96.     38
97.     39      6160 RUART=6160
98.     40      6161 WUART=6161
99.     41      6171 WTTY=6171
100.    42      6162 SKPDR=6162
101.    43      6163 SKPTBR=6163
102.    44      6165 WCRA=6165
103.    45      6175 WCRE=6175
104.    46      6174 WVR=6174
105.    47      6166 SFLAG1=6166
106.    48      6167 CFLAG1=6167
107.    49      6176 SFLAG3=6176
108.    50      6177 CFLAG3=6177
109.    51      6172 SKIP3=6172
110.    52      6173 SKIP4=6173
111.    53      /THE FOLLOWING DEFINE LOCATIONS OF VARIABLES
112.    54      /ALL VARIABLES ARE IN PAGE 0
113.    55      0020 TEMP=20    /TEMPORARY STORAGE
114.    56      0021 TEMP2=21    /MORE TEMPORARY STORAGE

```

```

115.    57    0023  TOTE=23      /DIGIT COUNT
116.    58    0024  WORD=24     /NUMBER READ BY COMMAND SCANNER
117.    59    0025  SCHAR=25    /CHAR INPUT BY COMMAND SCANNER
118.    60    0026  SPNTER=26   /POINTER INTO COMMAND TABLE
119.    61    0027  CAD=27      /CURRENT ADDRESS
120.    62    0030  CADF=30     /CURRENT FIELD
121.    63    0031  OCAIF=31    /OLD CADF VALUE
122.    64    0032  SHUT=32     /OPEN-SHUT FLAG (-1=SHUT,0=OPEN)
123.    65    0033  TRAD=33     /ADDRESS BREAKPOINT IS AT
124.    66    0034  TRADF=34    /FIELD BREAKPOINT IS IN
125.    67    0035  CONT=35     /ADDRESS TO CONTINUE FROM
126.    68    0036  IFSAVE=36   /FIELD TO CONTINUE WITH
127.    69    0037  INS=37      /INSTRUCTION TO BE SIMULATED
128.    70    0040  ADDR=40     /EFFECTIVE ADDRESS OF INS
129.    71    0041  DF1=41      /FIELD OF INS OPERAND
130.    72    0042  JADR=42     /PLACE FOR RAM PROGRAM TO RETURN TO
131.    73    0025  JADR1=25     /FUTURE JADR VALUE
132.    74    0043  GOADR=43    /ENTRY POINT TO RAM PROGRAM
133.    75    0044  NCONT=44    /NO OF TIMES TO CONTINUE PASS BKPT (ONES CM&LMT)
134.    76    0045  KEEP=45     /PLACE TO SAVE BREAKPOINT CONTENTS
135.    77    0046  CHKSUM=46   /PAPER TAPE CHECKSUM
136.    78          /THE FOLLOWING LOCATIONS MAY BE EXAMINED BY ODT-F USERS
137.    79    0047  ACSAVE=47    /SAVE AC
138.    80    0050  LSAVE=50     /SAVE LINK
139.    81    0051  MQSAVE=51    /SAVE MQ
140.    82    0052  DFSAVE=52   /SAVE DATA FIELD
141.    83    0053  MASK=53     /MASK FOR WORD SEARCH
142.    84    0054  LIMLO=54    /LOWER BOUND FOR WORD SEARCH
143.    85    0055  LIMHI=55    /UPPER BOUND FOR WORD SEARCH
144.    86          /THE SUBROUTINE LINKAGE LOCATIONS WILL LOOK LIKE
145.    87          /CALL1, 0      /LINKAGE TO PUSHJ ROUTINE
146.    88          /      JMP I .+1
147.    89          /      PUSHJ
148.    90          /RETRN1, PCPJ /LINKAGE TO POPJ ROUTINE
149.    91          /STACK, STACK1 /STACK POINTER
150.    92          / Q -- PUNCH CURRENT FIELD IN BIN FORMAT
151.    93          /ACTEMP, 0     /SAVE AC
152.    94    0056  CALL1=56
153.    95    0061  RETRN1=61
154.    96    0062  STACK=62
155.    97    0063  ACTEMP=63
156.    98          /THE BREAKPOINT RETURN LINKAGE LOCATIONS WILL LOOK LIKE
157.    99          /IN FIELD PROG TO DEBUG IS IN, AT ZPAT:
158.   100          /ZPAT, CIF F1A
159.   101          /      JMP ZPAT1
160.   102          /IN FIELD F1:
161.   103          /ZPAT1, JMP I .+1
162.   104          /      BKPT
163.   105    0064  ZPAT1=64
164.   106          /RAM PROGRAM LOCATIONS. THESE LOCATIONS ARE MODIFIED BY
165.   107          /GETCAD, SETCAD AND CDO ROUTINES AND THEN EXECUTED FOR
166.   108          /CHANGING FIELDS AND SIMULATING INSTRUCTION EXECUTION
167.   109          /DFSET1, 0     /CDF INS FOR MEMORY REFERENCE
168.   110          /NEWINS, 0    /INSTRUCTION TO SIMULATE
169.   111          /      SKP
170.   112          /      ISZ JADR/SIMULATE SKIP
171.   113          /IFSET, 0      /CIF INSTRUCTION
172.   114          /DFSET, 0     /CDF INSTRUCTION
173.   115          /      JMP I JADR /BACK TO RGM OR PROGRAM TO BE DEBUGGED
174.   116    0066  DFSET1=66
175.   117    0067  NEWINS=67

```

176.	118	0072	IFSET=72	
177.	119	0073	DFSET=73	
178.	120		/FIRST LOCATION OF RETURN ADDRESS STACK	
179.	121	0074	STACK1=74	
180.	122		PAUSE	
181.	123		/ODT-P VERSION 5 TAPE 2	
182.	124		/ ***** FIRST ROM PAGE	
183.	125	6000	*START	
184.	126		/ ***** INITIALIZE ROUTINE *****	
185.	127	6000 7200	INIT, CLA	
186.	128	6001 1245	TAD SUB1	/INIT SUBROUTINE LINKAGE
187.	129	6002 3057	DCA CALL1+1	
188.	130	6003 1246	TAD SUB2	
189.	131	6004 3060	DCA CALL1+2	
190.	132	6005 1247	TAD SUB3	
191.	133	6006 3061	DCA RETRN1	
192.	134	6007 1250	TAD STACK1	
193.	135	6010 3062	DCA STACK	
194.	136	6011 1251	TAD RAM1	/INIT RAM PROGRAM LOCATIONS
195.	137	6012 3070	DCA NEWINS+1	
196.	138	6013 1252	TAD RAM2	
197.	139	6014 3071	DCA NEWINS+2	
198.	140	6015 1253	TAD RAM3	
199.	141	6016 3074	DCA DFSET+1	
200.	142	6017 1254	TAD TRAP3	/INIT BKPT JUMP LINKAGE
201.	143	6020 3064	DCA ZPAT1	
202.	144	6021 1255	TAD TRAP4	
203.	145	6022 3065	DCA ZPAT1+1	
204.	146	6023 7040	CMA	/SFT DFSAVE TO "UNDEFINED" VALUE
205.	147	6024 3052	DCA DFSAVE	
206.	148	6025 3030	DCA CADF	/START IN FIELD 0
207.	149	6026 1256	TAD KCRA	
208.	150	6027 6165	WCRA	
209.	151	6030 7300	CIA CLL	
210.	152	6031 1257	TAD KCRE	
211.	153	6032 6175	WCRB	
212.	154	6033 7300	CIA CLL	
213.	155	6034 1260	TAD KTTY	
214.	156	6035 6171	WITY	
215.	157	6036 7300	CIA CLL	
216.	158	6037 3031	DCA QCADF	
217.	159	6040 3046	INIT1, DCA CHKSUM	/CLEAR CHECKSUM
218.	160	6041 4056	CALL	/TYPE CR,LF
219.	161	6042 6400	CRLF	
220.	162	6043 5644	JMP I PCREAD	/GO TO COMMAND SCANNER
221.	163		/ *** CONSTANTS	
222.	164	5044 6200	PCREAD, READ	
223.	165	6045 5460	SUB1, JMP I CALL1+2	
224.	166	6046 6061	SUB2, PUSHJ	
225.	167	6047 6072	SUB3, POPJ	
226.	168	6050 0074	STACK1, STACK1	
227.	169	6051 7410	RAM1, SKP	
228.	170	6052 2042	RAM2, ISZ JADR	
229.	171	6053 5442	RAM3, JMP I JADR	
230.	172	6054 5465	TRAP3, JMP I ZPAT1+1	
231.	173	6055 7236	TRAP4, BKPT	
232.	174	6056 7200	KCRA, 7200	
233.	175	6057 1560	KCRE, 1560	
234.	176	6060 7600	KTTY, 7600	
235.	177		/ ***** CALL AND RETURN ROUTINES *****	
236.	178		/JMS WILL NOT WORK IN A ROM SINCE THE FIRST WORD OF THE	

```

237. 179 /SUBROUTINE CANNOT BE WRITTEN WITH THE RETURN ADDRESS.SSS.
238. 180 /HENCE, A SUBROUTINE IS CALLED BY
239. 181 /
240. 182 / CALL
241. 183 / <SUBROUTINE NAME>
242. 184 /AND RETURNED FROM BY
243. 185 /
244. 186 /WHERE
245. 187 4056 CALL=JMS CALL1 /CALL SUBROUTINE OPCODE
246. 188 5461 RETURN=JME I RETRN1 /RETURN FROM SUBROUTINE OPCODE
247. 189 /CALL THE FOLLOWING ROUTINES THROUGH THE SUBROUTINE
248. 190 /LINKAGE IN PAGE ZERO (SEE TAPE 1)
249. 191 /ROUTINE TO CALL A SUBROUTINE AND PUSH
250. 192 6061 3063 /RETURN ADDRESS ON STACK
251. 193 6062 2062 PUSHJ, DCA ACTEMP /SAVE AC
252. 194 6063 1056 ISZ STACK /UPDATE STACK PTR
253. 195 6064 7001 TAD CALL1 /GET USER RETURN ADDRESS
254. 196 6065 3462 IAC /INCREMENT PAST ARGUMENT
255. 197 6066 1456 DCA I STACK /PUT IT ON STACK
256. 198 6067 3056 TAD I CALL1 /GET USER ENTRY ADDRESS
257. 199 6070 1063 DCA CALL1 /PUT IT IN CALL
258. 200 6071 5456 TAD ACTEMP /RESTORE AC
259. 201 JMP I CALL1 /GO TO USER ROUTINE
260. 202 /ROUTINE TO RETURN FROM A SUBROUTINE, POPPING RETURN ADDRESS
261. 203 6072 3063 /OFF STACK
262. 204 6073 1462 POPJ, DCA ACTEMP /SAVE AC
263. 205 6074 3056 TAD I STACK /GET RETURN ADDRESS
264. 206 6075 7060 DCA CALL1 /SAVE IT IN CALL1
265. 207 6076 1062 CMA CML /-1 IN AC, COMPLEMENT L
266. 208 /TAD STACK /DECREMENT STACK PTR
267. 209 6077 3062 /{(CARRY RESTORES L)
268. 210 6100 1063 DCA STACK /RESTORE UPDATED STACK PTR.
269. 211 6101 5456 TAD ACTEMP /RESTORE AC
270. 212 JMP I CALL1 /RETURN
271. 213 / ***** PRINT SUBROUTINES *****
272. 214 /SUBROUTINE TO TYPE A CHARACTER
273. 215 6102 6161 /CHAR ASSUMED IN AC
274. 216 6103 6163 TYPE, WUART /OUTPUT CHAR TO TTY
275. 217 6104 5303 SKPTBR /READY FOR NEXT CHAR?
276. 218 6105 7200 JMP .-1 /LOOP IF NOT
277. 219 6106 5461 CLA /CLEAR AC
278. 220 RETURN /RETURN
279. 221 /SUBROUTINE TO PRINT A NUMBER
280. 222 6197 3020 /PRINT CONTENTS OF AC IN OCTAL FOLLOWED BY A SPACE
281. 223 6110 1334 PNUM, DCA TEMP /SAVE NUMBER
282. 224 6111 3023 TAD M4 /INITIALIZE COUNT OF DIGITS
283. 225 6112 1020 DCA TOTE /PRINTED
284. 226 6113 7004 TAD TEMP /GET BACK NUMBER
285. 227 /RAL /FIRST SHIFT INTO LINK
286. 228 6114 7004 /LOOP 4 TIMES -, PRINT A DIGIT
287. 229 6115 7006 PNUM2, RAL /SHIFT AC, L THREE LEFT
288. 230 6116 3020 RIL
289. 231 6117 1020 DCA TEMP /SAVE NUMBER
290. 232 6120 0335 TAD TEMP /GET IT BACK
291. 233 6121 1340 AND P7 /ISOLATE DIGIT
292. 234 6122 4056 TAD P260 /CONVERT TO ASCII
293. 6123 6102 CALL; TYPE /TYPE IT OUT
294. 235 6124 1020 TAD TEMP /GET BACK AC (NOTE L STILL SAME)
295. 236 6125 2023 ISZ TOTE /FOURTH ITERATION?
296. 237 6126 5314 JMP PNUM2 /LOOP IF NOT.
297. 238 /PRINT A SPACE

```

298.	239	6127	7200	CLA		
299.	240	6130	1337	TAD	P240	/GET ASCII CODE FOR SPACE
300.	241	6131	4056	CALL	TYPE	/TYPE IT OUT
301.		6132	6102			
302.	242	6133	5461	RETURN		/RETURN
303.	243			/ *** CONSTANTS		
304.	244	6134	7774	M4,	-4	
305.	245	6135	0007	P7,	7	
306.	246	6136	0007		7	
307.	247	6137	0240	P240,	240	
308.	248	6140	0260	P260,	260	
309.	249			PAUSE		
310.	250			/ODT-F VERSION 5 TAPE 3		
311.	251			/ ***** SECND ROM PAGE		
312.	252	6200		*START+200		
313.	253			/ ***** COMMAND SCANNER *****		
314.	254			/THE COMMAND SCANNER INPUTS A COMMAND OF THE FORM		
315.	255			/(<NUMBER><>)<><CHAR> (WHERE THE NUMBER IS OPTIONAL).		
316.	256			/IT STORES THE NUMBER IN "WORD" AND JUMPS TO THE		
317.	257			/ROUTINE ASSOCIATED WITH THE CHARACTER.		
318.	258			/INITIALIZE NUMBER SCANNER		
319.	259	6200	7200	READ,	CLA	
320.	260	6201	3024	DCA	WORD	/SET WORD TO ZERO
321.	261	6202	1333	TAD	SM5	
322.	262	6203	3023	DCA	TOTE	/SET TOTE TO -5
323.	263			/INPUT A CHAR		
324.	264	6204	6162	READ1,	SKPDR	/CHAR IN INPUT BUFFER?
325.	265	6205	5204	JMP	.-1	/LOOP IF NOT
326.	266	6206	7200	CLA;	RUART	/PUT CHAR INTO AC
327.		6207	6160			
328.	267	6210	0336	AND	K0377;	DCA SCHAR /SAVE CHAR
329.		6211	3025			
330.	268			/ECHO CHAR ON TTY		
331.	269	6212	1025	TAD	SCHAR	/RETRIEVE CHAR
332.	270	6213	4056	CALL		
333.	271	6214	6102	TYPE		/TYPE IT OUT
334.	272			/NUMBER SCANNER		
335.	273			/SEE IF SCHAR IS OCTAL DIGIT		
336.	274	6215	1025	TAD	SCHAR	/GET CHAR ("0"=260,"7"=267)
337.	275	6216	1332	TAD	SM270	/SUB 270 ("0"=-10,"7"=-1)
338.	276	6217	7500	SMA		/IF AC NOT NEG, THEN CHAR HAS
339.	277					/CODE GR THAN THAT OF DIGIT
340.	278	6220	5235	JMP	READ2	/SO GO TO CHAR HANDLER.
341.	279	6221	1334	TAD	SP10	/ADD 10 ("0"=0,"7"=7)
342.	280	6222	7510	SFA		/IF AC NEG, THEN CHAR HAS
343.	281					/CODE LESS THAN THAT OF DIGIT
344.	282	6223	5235	JMP	READ2	/SO GO TO CHAR HANDLER
345.	283			/ADD DIGIT TO PARTIAL NUMBER IN WORD		
346.	284	6224	3020	DCA	TEMP	/SAVE DIGIT
347.	285	6225	1024	TAD	WORD	/GET NUMBER SO FAR
348.	286	6226	7104	CLL	RAL	/SHIFT AC LEFT THREE BITS
349.	287	6227	7006	RTL		
350.	288	6230	1020	TAD	TEMP	/ADD IN NEW DIGIT
351.	289	6231	3024	DCA	WORD	/SAVE RESULT IN WORD
352.	290			/CHECK FOR TOO MANY DIGITS, RETURN TO CHAR RDR		
353.	291	6232	2023	ISZ	TOTE	/5 DIGITS TYPED?
354.	292	6233	5204	JMP	READ1	/NO - GET NEXT CHAR
355.	293	6234	5337	JMP	ERROR	/YES - GO TO ERROR
356.	294			/CHAR SCANNER		
357.	295			/FIND INPUT CHAR IN TABLE1 AND JUMP TO THE		
358.	296			/ASSOCIATED ROUTINE IN TABLE2		

```

359.      297      /INITIALIZE SEARCH LOOP
360.      298      READ2,  CLA
361.      299      6235  7200      TAD  BLIST      /GET PTR TO BEGIN OF TABLE1
362.      300      6236  1256      DCA  SPNTER     /INIT PTR INTO TABLE1
363.      301      /SEARCH LOOP
364.      302      6240  1426      READ3,  TAD  I SPNTER  /GET CHAR FROM TABLE1
365.      303      6241  2026      ISZ  SPNTER     /PCINT SPNTER AT NEXT CHAR
366.      304      6242  7510      SEA
367.      305
368.      306      6243  5337      JMP  ERROR     /CHAR NOT IN TABLE1 - ERROR
369.      307      6244  7041      CIA
370.      308      6245  1025      TAD  SCHAR     /NEGATE TABLE CHAR
371.      309      6246  7640      SZA  CLA       /AED INPUT CHAR
372.      310
373.      311      6247  5240      SZA  CLA       /SUM ZERO IF CHARS SAME, SKIP
374.      312
375.      313      6250  1026      JMP  READ3     /IF SO
376.      314
377.      315
378.      316      6251  1257      JMP  READ3     /OTHERWISE, CONTINUE LOOP
379.      317      6252  3020      /JUMP TO ASSOCIATED ROUTINE IN TABLE2
380.      318      6253  1420      TAD  SPNTER     /GET PTR INTO TABLE1
381.      319      6254  3020      DCA  TEMP     /SPNTR NOW POINTS ONE EAST
382.      320      6255  5420      TAD  LTABL     /MATCHING CHAR).
383.      321      6256  6260      DCA  TEMP     /CONVERT INTO PTR INTO TABLE2
384.      322      6257  0022      TAD  I TEMP    /PUT TABLE2 ENTRY
385.      323
386.      324
387.      325
388.      326
389.      327
390.      328      6260  0215      TAD  I TEMP    /INTO AC
391.      329      6261  0212      JMP  I TEMP    /JUMP TO LOC POINTED TO
392.      330      6262  0257      JMP  I TEMP    /BY TABLE ENTRY
393.      331      6263  0256      BLIST, TABLE1 /PCINTER TO BEGINNIG OF TABLE1
394.      332      6264  0301      LTABL, TABLE2-TABLE1-1 /CONSTANT TO GET CORRESPONDING
395.      333      6265  0315
396.      334      6266  0337
397.      335      6267  0336
398.      336      6270  0302
399.      337      6271  0307
400.      338      6272  0303
401.      339      6273  0327
402.      340      6274  0320
403.      341      6275  0305
404.      342      6276  0324
405.      343      6277  0321
406.      344      6300  0273
407.      345      6301  0314
408.      346      6302  7777
409.      347      6303  6616
410.      348      6304  6625
411.      349      6305  6601
412.      350      6306  6654
413.      351      6307  6727
414.      352      6310  6730
415.      353      6311  6715
416.      354      6312  6670
417.      355      6313  7215
418.      356      6314  7200
419.      357      6315  7000

TABLE1,  215  / CR
         212  / LF
         257  / /
         256  / .
         301  / A
         315  / M
         337  / -
         336  /
         302  / B
         307  / G
         303  / C
         327  / W
         320  / P
         305  / E
         324  / T
         321  / Q
         273  / ;
         314  / L
         -1  /TABLE FOLLOWED BY NEG NUMBER

TABLE2,  CRDO
         LFDC
         SLDC
         DCTDO
         ADD
         MDO
         BADO
         UADO
         BDO
         GDO
         CDO

```

```

420.    358    6316    7303           WDO
421.    359    6317    7502           PDO
422.    360    6320    7464           EDO
423.    361    6321    7471           TDO
424.    362    6322    7454           CDO
425.    363    6323    7451           SEMIDO
426.    364    6324    7576           BIN-2
427.    365
428.    366           6332    *START+332
429.    367
430.    368
431.    369           / *** CONSTANTS
432.    370    6332    7510    SM270,  -270
433.    371    6333    7773    SM5,    -5
434.    372    6334    0010    SP10,   10
435.    373    6335    0277    SP277, 277
436.    374    6336    0377    R0377, 0377;PAUSE
437.    375
438.    376           6337    *START+337
439.    377
440.    378           / ***** ERROR HANDLER *****
441.    379           /TYPE "?" AND RETURN TO COMMAND SCANNER
442.    380    6337    7200    ERRCR,  CLA
443.    381    6340    1335           TAD SP277    /GET QUESTION MARK
444.    382    6341    4056           CALL
445.    383    6342    6102           TYPE        /TYPE IT OUT
446.    384    6343    4056           CALL
447.    385    6344    6400           CRLF       /TYPE CR,LF
448.    386    6345    5200           JMP READ
449.    387
450.    388
451.    389
452.    390
453.    391           /ODT-F VERSION 5 TAPE 4
454.    392           / ***** THIRD ROM PAGE
455.    393           6400    *START+400
456.    394           / ***** EXAMINE DEPOSIT ROUTINES *****
457.    395           /SUBROUTINE TO TYPE CR, LF AND SHUT REGISTER
458.    396    6400    1257    CRIF,   TAD TP215    /GET CR
459.    397    6401    4056           CALL
460.    398    6402    6102           TYPE        /TYPE IT
461.    399    6403    1256           TAD TP212    /GET LF
462.    400    6404    4056           CALL
463.    401    6405    6102           TYPE        /TYPE IT
464.    402    6406    7040           CMA        /SFT AC TO -1
465.    403    6407    3032           DCA SHUT    /STORE IN SHUT
466.    404    6410    5461           RETURN     /RETURN
467.    405           /SUBROUTINE TO CLOSE REG
468.    406           /SEE IF REG ALREADY SHUT
469.    407    6411    2032    CLOSE,  ISZ SHUT    /SKIP IF SHUT=-1
470.    408    6412    7410           SKP
471.    409    6413    5461           RETURN     /RETURN IF REG ALREADY SHUT
472.    410           /SEE IF VALUE TYPED IN
473.    411    6414    4056           CALL
474.    412    6415    6423           NTPED      /SEE IF NUMBER TYPED
475.    413    6416    5461           RETURN     /RETURN IF NOT
476.    414           /STORE TYPED VALUE IN REG
477.    415    6417    1024           TAD WORD    /GET NEW VALUE
478.    416    6420    4056           CALL
479.    417    6421    6442           SETCAD     /STORE IT IN CAD
480.    418    6422    5461           RETURN     /RETURN

```



```

481.      419      /SUBROUTINE TO SEE IF NUMBER HAS BEEN TYPED
482.      420      /SKIP ON RETURN IF ANY NUMBER HAS BEEN TYPED SINCE
483.      421      /LAST COMMAND
484.      422      6423 1023 NNTYPED, TAD TOTE      /TOTE IS -5 INITIALLY, AND
485.      423      6424 7041 CIA      / INCREMENTED ONLY IF A NUMBER
486.      424      6425 1255 TAD TM5      / IS TYPED
487.      425      6426 7640 SZA CLA      /SKIP IF TOTE=-5
488.      426      6427 2462 ISZ I STACK /INCREMENT RETURN ADDR
489.      427      6430 5461 RETURN
490.      428      /SUBROUTINE TO GET CONTENTS OF CURRENT LOCATION
491.      429      /NEW CONTENTS RETURNED IN AC
492.      430      6431 1030 GETCAD, TAD CADF      /GET FIELD OF CURRENT LOC
493.      431      6432 1260 TAD TP6201 /ADD IN "CEFF" INSTRUCTION
494.      432      6433 3073 DCA DFSET /STORE IT IN RAM PROGRAM
495.      433      6434 1261 TAD PGTCDD1 /SET RAM PROG TO RETURN TO GETCD1
496.      434      6435 3042 DCA JADR
497.      435      6436 5073 JMP DFSET /EXECUTE RAM PROGRAM - CHANGE FIELD
498.      436      /TO THAT OF CURRENT LOCATION
499.      437      6437 1427 GETCD1, TAD I CAD /GET CONTENTS OF CURRENT LOC
500.      438      6440 6201 CDF F1A /RESTORE DATA FIELD
501.      439      6441 5461 RETURN /RETURN
502.      440      /SUBROUTINE TO SET CONTENTS OF CURRENT LOCATION
503.      441      /NEW CONTENTS PASSED IN AC
504.      442      6442 3020 SETCAD, DCA TEMP /SAVE AC
505.      443      6443 1030 TAD CADF /GET FIELD OF CURRENT LOC
506.      444      6444 1260 TAD TP6201 /ADD IN "CDF" INSTRUCTION
507.      445      6445 3073 DCA DFSET /STORE IT IN RAM
508.      446      6446 1262 TAD PSTCD1 /SET RAM PRG TO RETURN TO SETCD1
509.      447      6447 3042 DCA JADR
510.      448      6450 5073 JMP DFSET /EXECUTE RAM PROGRAM - CHANGE FIELD
511.      449      /TO THAT OF CURRENT LOCATION
512.      450      6451 1020 SETCD1, TAD TEMP /RESTORE AC
513.      451      6452 3427 DCA I CAD /SET CURRENT LOC TO NEW VALUE
514.      452      6453 6201 CDF F1A /RESTORE DATA FIELD
515.      453      6454 5461 RETURN /RETURN
516.      454      / *** CONSTANTS
517.      455      6455 7773 TM5, -5
518.      456      6456 0212 TP212, 212
519.      457      6457 0215 TP215, 215
520.      458      6460 6201 TP6201, 6201
521.      459      6461 6437 PGTCDD1, GETCD1
522.      460      6462 6451 PSTCD1, SETCD1
523.      461      6463 6172 LISN, SKIP3 / THE TTY LISN ROUTINE FOR THE BIN
524.      462      6464 7000 NCP / LOADER
525.      463      6465 6166 SELAG1 / SET READER RUN
526.      464      6466 6172 SKIP3
527.      465      6467 5266 JMP .-1
528.      466      6470 6167 CFLAG1
529.      467      6471 6162 SKPDR
530.      468      6472 5271 JMP .-1
531.      469      6473 7200 CIA
532.      470      6474 6160 BUART
533.      471      6475 0277 AND TTYM
534.      472      6476 5461 RETURN
535.      473      6477 0377 TTYM, 0377
536.      474
537.      475      /ODT-F VERSION 5 TAPE 5
538.      476      / ***** EXAMINE/DEPOSIT ROUTINES - CONTINUED *****
539.      477      / ***** FCURTH ROM PAGE
540.      478      6600 *START+600
541.      479      /TRANSFER ADDRESS TO COMMAND AND SCANNER

```

```

542. 480 6600 6200 COMM, READ
543. 481 /SLASH HANDLER
544. 482 /OPEN LOCATION "CADF"."CAD"
545. 483 /IF NUMBER TYPED, SET CAE
546. 484 6601 1031 SLDC, TAD OCADF
547. 485 6602 3030 DCA CADF
548. 486 6603 4056 CALL
549. 487 6604 6423 NTYPEP /SEE IF NUMBER TYPED
550. 488 6605 5210 JMP SLDO1 /NO NUM TYPED, LEAVE CAD ALONE
551. 489 6606 1024 TAD WORD /ELSE GET NUM TYPED
552. 490 6607 3027 DCA CAD /AND SET CAD TO IT
553. 491 /TYPE CONTENTS OF "CADF"."CAD"
554. 492 6610 3032 SLDO1, DCA SHUT /SET REGISTER STATUS TO "OPEN"
555. 493 6611 4056 CALL
556. 494 6612 6431 GETCAD /GET CONTENTS OF CURRENT LOC
557. 495 6613 4056 CALL
558. 496 6614 6107 ENUM /TYPE IT OUT
559. 497 6615 5600 JMP I COMM /RETURN TO COMMAND SCANNER
560. 498 /CR HANDLER
561. 499 /CLOSE LOCATION
562. 500 6616 4056 CRDC, CALL
563. 501 6617 6411 CLOSE /CLOSE LOCATION
564. 502 6620 1031 TAD OCADF
565. 503 6621 3030 DCA CADF
566. 504 6622 4056 CALL
567. 505 6623 6400 CRLF /TYPE CR, LF
568. 506 6624 5600 JMP I COMM /GET NEXT COMMAND
569. 507 /LF HANDLER
570. 508 /CLOSE LOCATION AND OPEN NEXT LOCATION
571. 509 6625 4056 LFDO, CALL
572. 510 6626 6411 CLOSE /CLOSE LOCATION
573. 511 6627 1344 TAD SP215 /GET CR
574. 512 6630 4056 CALL
575. 513 6631 6102 TYPE /TYPE IT
576. 514 6632 4056 CALL
577. 515 6633 6102 TYPE /TYPE A NULL (GIVES TIME FOR CR)
578. 516 6634 2027 ISZ CAD /POINT CAD AT NEXT LOC
579. 517 6635 7000 NOP /IN CASE ISZ SKIPS
580. 518 6636 1030 TAD CADF /COMPARE CURRENT AND SAVED FIELDS
581. 519 6637 7041 CIA
582. 520 6640 1031 TAD OCADF
583. 521 6641 7650 SNA CLA /DIFFERENT?
584. 522 6642 5245 JMP LFDO1 /NO, GO ON
585. 523 6643 4056 CALL /YES - PRINT FIELD
586. 524 6644 7347 PFIELD
587. 525 /THE FOLLOWING IS AN ENTRY POINT FOR ANY ROUTINE
588. 526 /THAT TYPES OUT WHAT LOCATION IT IS OPENING.
589. 527 6645 1027 LFDO1, TAD CAD /GET CAD
590. 528 6646 4056 CALL
591. 529 6647 6107 PNUM /TYPE IT OUT
592. 530 6650 1345 TAD SP257 /GET ASCII FOR "/"
593. 531 6651 4056 CALL
594. 532 6652 6102 TYPE /TYPE IT OUT
595. 533 6653 5210 JMP SLDO1 /REST IS LIKE OPEN REG ROUTINE
596. 534 /. HANDLER
597. 535 /SET FIELD OF CURRENT LOCATION
598. 536 6654 1024 DOTDO, TAD WORD /GET FIELD SPEC FROM WORD
599. 537 6655 7104 CLL RAL /FIELD SPEC INTO BITS 6-8
600. 538 6656 7006 RTL
601. 539 6657 3030 DCA CADF /STORE IT AS CURRENT FIELD
602. 540 6660 1030 TAD CADF

```

```

603.    541    6661  3031          DCA OCADF
604.    542    6662  1052          TAD DFSAVE          /DFSAVE INITIALIZED?
605.    543    6663  7700          SMA CLA
606.    544    6664  5600          JMP I COMM          /YES - RETURN
607.    545    6665  1030          TAD CADF            /NO - INIT DFSAVE TO CADF
608.    546    6666  3052          DCA DFSAVE
609.    547    6667  5600          JMP I COMM          /GET NEXT COMMAND
610.    548
611.    549          / HANDLER
612.    550          /PRETEND CAD IS A MEMORY REFERENCE INS AND OPEN
613.    551    6670  4056          /LOCATION REFERENCED. IGNORE I BIT.
614.    552    6671  6411          UADO, CALL
615.    553    6672  4056          CLOSE              /CLOSE LOCATION
616.    554    6673  6400          CALF                /TYPE CR,LF
617.    555    6674  4056          CALL
618.    556    6675  6431          GETCAD              /GET CONTENTS
619.    557    6676  3020          DCA TEMP            /SAVE THEM IN TEMP
620.    558    6677  1031          TAD OCADF
621.    559    6700  3030          DCA CADF
622.    560    6701  1020          TAD TEMP            /GET BACK CONTENTS
623.    561    6702  0342          AND SP177           /ISCLATE PAGE ADDR BITS
624.    562    6703  3021          DCA TEMP2           /AND SAVE THEL IN TEMP2
625.    563    6704  1020          TAD TEMP            /GET CONTENTS AGAIN
626.    564    6705  0343          AND SP200           /ISOLATE PAGE ZERO BIT
627.    565    6706  7650          SNA CLA             /REFERENCE TO PAGE ZERC?
628.    566    6707  5312          JMP UAD01           /YES - SKIF NEXT CODE
629.    567    6710  1027          TAD CAD             /NO - GET CURRENT ADR
630.    568    6711  0351          AND SP7600          /ISOLATE PAGE NUMBER
631.    569    6712  1021          UAD01, TAD TEMP2    /ADD IN PAGE ADDR
632.    570    6713  3027          DCA CAD             /PUT INTO CAD
633.    571    6714  5245          JMP LFD01           /REST LIKE LF
634.    572
635.    573          / HANDLEF
636.    574    6715  4056          /OPEN LOC PCINTED AT BY CURRENT LOC
637.    575    6716  6411          BADC, CALL
638.    576    6717  4056          CLOSE              /CLOSE CURRENT LOC
639.    577    6720  6400          CALL
640.    578    6721  4056          CRLF                /TYPE CR,LF
641.    579    6722  6431          CALL
642.    580    6723  3027          GETCAD              /GET CONTENTS OF CURRENT LOC
643.    581    6724  1031          DCA CAD             /MAKE IT INTO NEW LOC
644.    582    6725  3030          TAD OCADF
645.    583    6726  5245          DCA CADF
646.    584          JMP LFD01           /REST LIKE LF
647.    585
648.    586    6727  1337          / A,M,I HANDLER
649.    587    6730  1340          /OPEN LOC CONTAINING AC, MASK, OR INSTRUCTION FIELD
650.    588    6731  3027          ADO, TAD ADDR1      /GET REGISTER ADDRESS - ACSAVE
651.    589    6732  1030          MDO, TAD ADDR2      /OF MASK
652.    590    6733  3031          DCA CAD             /PUT INTO CAD
653.    591    6734  1341          TAD ADDR4
654.    592    6735  3030          DCA CADF
655.    593    6736  5210          JMP SLDO1           /REST LIKE SLASH
656.    594    6737  7774          ADDR1, ACSAVE-MASK
657.    595    6740  0053          ADDR2, MASK
658.    596    6741  0000          ADDR4, F1A
659.    597
660.    598    6742  0177          / *** CONSTANTS
661.    599    6743  0200          SP177, 177
662.    600    6744  0215          SP200, 200
663.    601    6745  0257          SP215, 215
664.    601    6745  0257          SP257, 257

```

664.	602	6746	0260	SP260, 260
665.	603	6747	0400	SP400, 400
666.	604	6750	6201	SP6201, 6201
667.	605	6751	7600	SP7600, 7600
668.	606			PAUSE
669.	607			/ODT-F VERSION 5 TAPE 6
670.	608			/ ***** BREAKPOINT/CONTINUE ROUTINES *****
671.	609			/ ***** FIFTH ROM PAGE
672.	610	7000		*START+1000
673.	611			/ C HANDLER
674.	612			/CONTINUE EXECUTION OF PROGRAM
675.	613	7000	1031	CDO, TAD OCAF
676.	614	7001	3030	DCA CADF
677.	615	7002	4056	CALL
678.	616	7003	6400	CELF
679.	617	7004	1024	TAD WORD /SET CONTINUE COUNT TO
680.	618	7005	7040	CMA / ITERATE PAST BREAKPOINT
681.	619	7006	3044	DCA NCONT / SPECIFIED NUMBER OF TIMES
682.	620			/SIMULATE EXECUTION OF THE INSTRUCTION IN
683.	621			/LOCATION CONTF.CONT
684.	622			/HANDLE ICT OR OPERATE INSTRUCTION
685.	623	7007	1035	CONTO, TAD CONT /SET CADF.CAD TO CONTF.CONT
686.	624	7010	3027	DCA CAD
687.	625	7011	1036	TAD IFSAVE /IFSAVE IS CONTINUE FIELD
688.	626	7012	3030	DCA CADF
689.	627	7013	4056	CALL /GET CONTINUE INSTRUCTION
690.	628	7014	6431	GETCAD
691.	629	7015	3037	DCA INS /STORE IT IN INS
692.	630	7016	2035	ISZ CONT /POINT CONT AT NEXT INS TO EXECUTE
693.	631	7017	1035	TAD CONT /SET ADDR TO CONT PROG EXEC FROM
694.	632	7020	3025	DCA JADR1
695.	633	7021	1037	TAD INS
696.	634	7022	7100	CLL
697.	635	7023	1346	TAD FP2000 /OVERFLOW SETS L FOR ICT OR OPER
698.	636	7024	7620	SNL CLA
699.	637	7025	5233	JMP CONT1 /JMP IF NOT IOT OR OPER
700.	638	7026	1037	TAD INS /NEWINS INS
701.	639	7027	3067	DCA NEWINS
702.	640	7030	1334	TAD PNWINS /SET RAM PROGRAM EXECUTION TO
703.	641	7031	3043	DCA GOADR /BEGIN FROM "NEWINS"
704.	642	7032	5740	JMF I PCNT5 /GO TO EXECUTE RAM
705.	643			/PUT EFFECTIVE ADDRESS OF MEMORY REFERENCE INSTRUCTION
706.	644			/INTO ADDR
707.	645	7033	1037	CONT1, TAD INS
708.	646	7034	0343	AND FP177 /GET ADDRESS ON PAGE
709.	647	7035	3040	DCA ADDR
710.	648	7036	1037	TAD INS /TEST FOR ZERO PAGE
711.	649	7037	0344	AND FP200
712.	650	7040	7650	SNA CLA
713.	651	7041	5247	JMP CONT2 /ZERO PAGE - GO ON
714.	652	7042	7040	CMA /CURRENT PAGE - GET INS LOCATION
715.	653	7043	1035	TAD CONT /CENT POINTS ONE PAST INS LOC
716.	654	7044	0351	AND FP7600 /ISOLATE PAGE NUMBER
717.	655	7045	1040	TAD ADDR /ADD TO GET
718.	656	7046	3040	DCA ADDR /NEW EFFECTIVE ADDRESS
719.	657			/HANDLE INDIRECT REPERENCE
720.	658	7047	1036	CONT2, TAD IFSAVE /ASSUME EA IS IN INSTRUCTION FIELD
721.	659	7050	3041	DCA DF1 / SO SET DF1 IFSAVE
722.	660	7051	1037	TAD INS /TEST INS FOR INDIRECT
723.	661	7052	0345	AND FP400
724.	662	7053	7650	SNA CLA

725.	663	7054	5276	JMP CONT3	/DIRECT - GO ON
726.	664	7055	1052	TAD DFSAVE	/INDIRECT - SO EA IS IN DATA FIELD
727.	665	7056	3041	DCA DF1	/ SO SDT DF1 DFSAVE
728.	666	7057	1040	TAD ADDR	/SET CAD ADDR
729.	667	7060	3027	DCA CAD	
730.	668			/HANDLE AUTO-INCREMENT	
731.	669	7061	1040	TAD ADDR	/SEE IF ADDR IS AUTO-INC REGISTER
732.	670	7062	0352	AND FP7770	
733.	671	7063	1342	TAD FM10	
734.	672	7064	7640	SZA CLA	
735.	673	7065	5273	JMP CONT2A	/NCT AUTO-INCR - GO ON
736.	674	7066	4056	CALL	/AUTO-INCR - GET REGISTER VALUE
737.	675	7067	6431	GETCAD	
738.	676	7070	7001	IAC	/INCREMENT IT
739.	677	7071	4056	CALL	/AND RESTORE
740.	678	7072	6442	SETCAD	
741.	679			/FINISH HANDLING INDIRECT	
742.	680	7073	4056	CONT2A, CALL	/GET CONTENTS OF ADDF
743.	681	7074	6431	GETCAD	
744.	682	7075	3040	DCA ADDR	/AND MAKE THAT NEW ADDR
745.	683			/HANDLE JMS, JMP INSTRUCTIONS	
746.	684	7076	1037	CONT3, TAD INS	/SEE IF INS IS JMP OR JMS
747.	685	7077	7100	CIL	
748.	686	7100	0350	AND FP7000	
749.	687	7101	1347	TAD FP4000	
750.	688	7102	7420	SNL	/L=1 IF INS IS JMS OR JMP
751.	689	7103	5321	JMP CONT4	/NCT JMS OR JMP - GO ON
752.	690	7104	7640	SZA CLA	/JMS OR JMP - AC=0 IF INS IS JMS
753.	691	7105	5314	JMP CONT3A	/NCT JMS - GO ON
754.	692	7106	1040	TAD ADDR	/JMS - EMULATE JMS
755.	693	7107	3027	DCA CAD	
756.	694	7110	1035	TAD CONT	
757.	695	7111	4056	CALL	
758.	696	7112	6442	SETCAD	/SET RETURNBURN ADDRESS
759.	697	7113	2040	ISZ ADDR	/POINT ADDR AT SUBROUTINE BODY
760.	698	7114	1040	CONT3A, TAD ADDR	/GET JUMP DESTINATION
761.	699	7115	3025	DCA JADR1	/AND SET IT
762.	700	7116	1335	TAD PIFSET	/START RAM PROG EXECUTION
763.	701	7117	3043	DCA GOADR	/FROM IFSET
764.	702	7120	5740	JME I PCNT5	/GO TO EXECUTE RAM PROG
765.	703			/HANDLE AND, TAD, DCA, ISZ INSTRUCTIONS	
766.	704	7121	7200	CONT4, CLA	
767.	705	7122	1037	TAD INS	/GET INS
768.	706	7123	0350	AND FP7000	/ISOLATE OPCODE
769.	707	7124	1337	TAD IADDR	
770.	708	7125	3067	DCA NEWINS	/SET NEWINS OPCODE I ADDR
771.	709	7126	1041	TAD DF1	
772.	710	7127	1341	TAD CDF0	
773.	711	7130	3066	DCA DFSET1	/SET DFSET1 CDF DF1
774.	712	7131	1336	TAD PDFST1	/SET RAM PROG EXECUTION
775.	713	7132	3043	DCA GOADR	/ TO BEGIN FROM DFSET1
776.	714	7133	5740	JME I PCNT5	/GO TO EXECUTE RAM PROG
777.	715			/ *** CONSTANTS	
778.	716	7134	0067	PNWINS, NEWINS	
779.	717	7135	0072	PIFSET, IFSET	
780.	718	7136	0066	PDFST1, DFSET1	
781.	719	7137	0440	IADDR, 400 ADDR	
782.	720	7140	7400	PCNT5, CCNT5	
783.	721	7141	6201	CDF0, CEF	
784.	722	7142	7770	FM10, -10	
785.	723	7143	0177	FP177, 177	

786.	724	7144	0200	FP200, 200
787.	725	7145	0400	FP400, 400
788.	726	7146	2000	FP2000200C, 2000
789.	727	7147	4000	FP4000, 4000
790.	728	7150	7000	FP7000, 7000
791.	729	7151	7600	FP7600, 7600
792.	730	7152	7770	FP7770, 7770
793.	731			PAUSE
794.	732			/ODT-F VERSION 5 TAPE 7
795.	733			/ ***** BREAKPOINT/CONTINUE ROUTINES - CONTINUED *****
796.	734			/ ***** SIXTH ROM PAGE
797.	735	7200		*START+1200
798.	736			/ G HANDLER
799.	737			/GO TO A SPECIFIED LOCATION
800.	738	7200	1031	GDO, TAD OCADF
801.	739	7201	3030	DCA CADF
802.	740	7202	4056	CALL
803.	741	7203	6400	CRLF
804.	742	7204	7040	CMA /SET CONTINUE COUNT TO ZERO
805.	743	7205	3044	DCA NCONT
806.	744	7206	1024	TAD WORD /SET ADDR TO CONTINUE PROGRAM
807.	745	7207	3025	DCA JADR1 / EXECUTION FROM
808.	746	7210	1030	TAD CADF /SET FIELD TO CONTINUE PROGRAM
809.	747	7211	3036	DCA IFSAVE / EXECUTION FROM
810.	748	7212	1365	TAD PIFSTA /SET RAM PROGRAM EXECUTION TO
811.	749	7213	3043	DCA GOADR / BEGIN AT HFSET
812.	750	7214	5767	JME I PCONT5 /GO TO EXECUTE RAM PROGRAM
813.	751			/ B HANDLER
814.	752			/SET BREAKPOINT AT A SPECIFIED LOCATION
815.	753	7215	1031	BDO, TAD OCADF
816.	754	7216	3030	DCA CADF
817.	755	7217	4056	CALL /TYPE CR,LF
818.	756	7220	6400	CRLF
819.	757	7221	4056	CALL /SEE IF VALUE SPECIFIED
820.	758	7222	6423	NTYPED
821.	759	7223	5231	JMP BDO1 /VALTE SPECIFIED - GO ON
822.	760	7224	1024	TAD WORD /SET BREAKPOINT ADDRESS
823.	761	7225	3033	DCA TRAD
824.	762	7226	1030	TAD CADF /AND FIELD
825.	763	7227	3034	DCA TRADF
826.	764	7230	5770	JME I PREAD /GET NEXT COMMAND
827.	765	7231	1371	BDO1, TAD ODTLOC /NO VALUE SPECIFIED -
828.	766	7232	3033	DCA TRAD / CLEAR BREAKPOINT BY SETTING
829.	767	7233	1000	TAD F1A / IT WITHIN ODT-F
830.	768	7234	3034	DCA TRADF
831.	769	7235	5770	JMP I PREAD /GET NEW COMMAND
832.	770			/BREAKPOINT RETURN
833.	771			/ENTRY TO ODT UPON ENCOUNTERING BREAKPOINT
834.	772			/SAVE STUFF THEN GO TO COMMAND SCANNER
835.	773	7236	3047	BKPT, DCA ACSAVE /SAVE AC
836.	774	7237	7004	FAL /SAVE LINK
837.	775	7240	3050	DCA LSAVE
838.	776	7241	7701	ACL /SAVE MQ
839.	777	7242	3051	ECA MQSAVE
840.	778	7243	6214	RDF /SAVE PROGRAM DATA FIELD
841.	779	7244	3052	DCA DFSAVE
842.	780	7245	6201	CDF F1A /SET ODT-F F-P DATA FIELD
843.	781	7246	1034	TAD TRADF /SAVE PROGRAM INSTRUCTION FIELD
844.	782	7247	3036	DCA IFSAVE
845.	783	7250	1033	TAD TRAD /AND BREAKPOINT LOCATION
846.	784	7251	3035	DCA CONT /FOR CONTINUE

847.	785	7252	1033	TAD TRAD	/RESTORE BREAKPOINT LOCATION
848.	786	7253	3027	DCA CAD	/CONTENTS
849.	787	7254	1034	TAD TRADF	
850.	788	7255	3030	DCA CADF	
851.	789	7256	1045	TAD KEEP	
952.	790	7257	4056	CALL	
853.	791	7260	6442	SETCAD	
854.	792	7261	1030	TAD CADF	/INITIALIZE OCADF
855.	793	7262	3031	DCA OCADF	
856.	794	7263	2044	ISZ NCONT	/CONTINUE COUNT OVER?
857.	795	7264	5766	JMP I PCONTO	/NC - CONTINUE
858.	796	7265	4056	CALL	/TYPE OUT BKPT FIELD
959.	797	7266	7347	FFIELD	
860.	798	7267	1033	TAD TRAD	/AND LOCATION
861.	799	7270	4056	CALL	
862.	800	7271	6107	FNUM	
863.	801	7272	1361	TAD AP250	/TYPE "("
864.	802	7273	4056	CALL	
865.	803	7274	6102	TYPE	
866.	804	7275	1047	TAD ACSAVE	/TYPE OUT AC
867.	805	7276	4056	CALL	
868.	806	7277	6107	PNUM	
869.	807	7300	4056	CALL	/TYPE CR, LF
870.	808	7301	6400	CRLF	
871.	809	7302	5770	JMP I PREAD	/GET NEXT COMMAND
872.	810				/ ***** WCRD SEARCH ROUTINE END *****
873.	811				/ W HANDLER
874.	812				/SEARCH BETWEEN LIMLO AND LIMHI FOR WORDS THAT
875.	813				/MATCH SPECIFIED NUMBER IN MASKED BITS
876.	814	7303	1031	WDO, TAD OCADF	
877.	815	7304	3030	DCA CADF	
878.	816	7305	4056	CALL	/TYPE CR,LF
879.	817	7306	6400	CRLF	
890.	818	7307	1024	TAD WORD	/MASK SPECIFIED NUMBER
881.	819	7310	0053	AND MASK	
882.	820	7311	3024	DCA WORD	
883.	821	7312	1054	TAD LIMLO	/INITIALIZE CAD AS MEMORY
884.	822	7313	3027	DCA CAD	/ POINTER FOR SEARCH
885.	823	7314	4056	WDO1, CALL	/GET MEMORY WORD
886.	824	7315	6431	GETCAD	
887.	825	7316	0053	AND MASK	/MASK IT
888.	826	7317	7041	CIA	/CCMPARE TO WORD
889.	827	7320	1024	TAD WORD	
890.	828	7321	7640	SZA CLA	/EQUAL?
891.	829	7322	5337	JMP WDO2	/NC ONO - GET NEXT
892.	830	7323	1027	TAD CAD	/YES - PRINT LOC
893.	831	7324	4056	CALL	
894.	832	7325	6107	FNUM	
895.	833	7326	1363	TAD AP257	/PRINT "/"
896.	834	7327	4056	CALL	
897.	835	7330	6102	TYPE	
898.	836	7331	4056	CALL	/PRINT CONTENTS
899.	837	7332	6431	GETCAD	
900.	838	7333	4056	CALL	
901.	839	7334	6107	PNUM	
902.	840	7335	4056	CALL	/TYPE CR, IF
903.	841	7336	6400	CRLF	
904.	842	7337	1027	WDO2, TAD CAD	/GET LOC
905.	843	7340	2027	ISZ CAD	/POINT AT NEXT LOC
906.	844	7341	7000	NCP	/IN CASE ISZ SKIPS
907.	845	7342	7041	CIA	/CCMPARE LOC WITH LIMHI

```

908. 846 7343 1055 TAD LIMHI
909. 847 7344 7640 SZA CLA /LIMHI REACHED?
910. 848 7345 5314 JMP WDO1 /NO - REPEAT LOOP
911. 849 7346 5770 JMP I PREAD /YES - GET NEXT COMMAND
912. 850 / ***** PRINT FIELD SUBROUTINE *****
913. 851 /SUBROUTINE TO PRINT FIELD
914. 852 /PRINT FIELD SPECIFIED BY CADP FOLLOWED BY "."
915. 853 7347 1030 PFIELD, TAC CADF /GET FIELD
916. 854 7350 7110 CLL RAR /PUT IT IN BITS 9-11
917. 855 7351 7012 RIR
918. 856 7352 1364 TAD AP260 /AID IN ASCII FOR NUMBER
919. 857 7353 4056 CALL /TYPE IT OUT
920. 858 7354 6102 TYPE
921. 859 7355 1362 TAD AP256 /TYPE "."
922. 860 7356 4056 CALL
923. 861 7357 6102 TYPE
924. 862 7360 5461 RETURN /RETURN
925. 863 / *** CONSTANTS
926. 864 7361 0250 AP250, 250
927. 865 7362 0256 AP256, 256
928. 866 7363 0257 AP257, 257
929. 867 7364 0260 AP260, 260
930. 868 7365 0072 PIFSTA, IFSSET
931. 869 7366 7007 PCONT0, CONT0
932. 870 7367 7400 PCONT5, CCNT5
933. 871 7370 6200 PREAD, READ
934. 872 7371 0620 ODTLOC, 20 /ANY UNEXECUTED LOC IN ODT-F
935. 873 PAUSE
936. 874 /ODT-F VERSION 5 TAPE 8
937. 875 / ***** SEVENTH ROM PAGE
938. 876 7400 *START+1400
939. 877 / ***** BREAKPOINT/CONTINUE ROUTINES - CONTINUED *****
940. 878 /RESTORE CONDITIONS OF PROGRAM TO BE DEBUGGED AND
941. 879 /EXECUTE RAM PROGRAM
942. 880 /RAM PROGRAM RETURNS TO PROGRAM BEING DEBUGGED.
943. 881 7400 7200 CONTS, CIA
944. 882 7401 1033 TAD TRAD /SET CADF.CAD TRADF.TRAD
945. 883 7402 3027 DCA CAD
946. 884 7403 1034 TAD TRADF
947. 885 7404 3030 DCA CADF
948. 886 7405 4056 CALL /GET BREAKPOINT CONTENTS
949. 887 7406 6431 GETCAD
950. 888 7407 3045 DCA KEEP /AND SAVE THEM IN KEEP
951. 889 7410 1247 TAD TRAP /PUT TRAP INSTRUCTION
952. 890 7411 4056 CALL
953. 891 7412 6442 SETCAD /INTO BREAKPOINT LOCATION
954. 892 7413 1250 TAD PZPAT /POINT CAD AT BKPT LINKAGEEGE
955. 893 7414 3027 DCA CAD / LOCATIONS
956. 894 7415 1243 TAD TRAP1 /SET BREAKPOINT LINKAGE LOCATIONS
957. 895 7416 4056 CALL
958. 896 7417 6442 SETCAD
959. 897 7420 2027 ISZ CAD
960. 898 7421 1244 TAD TRAP2
961. 899 7422 4056 CALL
962. 900 7423 6442 SETCAD
963. 901 7424 1036 TAD IFSAVE /SET IFSSET CIF IFSAVE
964. 902 7425 1245 TAD PBCIF
965. 903 7426 3072 DCA IFSSET
966. 904 7427 1052 TAD DFSAVE /SET DFSSET CDF DFSAVE
967. 905 7430 1246 TAD PBCDF
968. 906 7431 3073 DCA DFSSET

```


969.	907	7432	1025	TAD JADR1	
970.	908	7433	3042	DCA JADR	
971.	909	7434	1051	TAD MSAVE	/RESTORE MC
972.	910	7435	7421	MCL	
973.	911	7436	1050	TAD LSAVE	/RESTORE LINK
974.	912	7437	7010	RAR	
975.	913	7440	7200	CLA	/RESTORE AC
976.	914	7441	1047	TAD ACSAVE	
977.	915	7442	5443	JMP I GOADR	/GC TO EXECUTE RAM PROGRAM
978.	916			/ *** CONSTANTS	
979.	917	7443	6202	TRAP1, CIF FIA	
980.	918	7444	5064	TRAF2, JMF ZPAT1	
981.	919	7445	6202	PBCIF, CIF	
982.	920	7446	6201	PBCDF, CDF	
983.	921	7447	5005	TRAP, JMF ZFAT	
984.	922	7450	0005	PZPAT, ZFAT	
985.	923			/ ***** PUNCH ROUTINES *****	
986.	924			/ ; HANDLER	
987.	925			/SET LOWER LIMIT OF PUNCH COMMAND	
988.	926	7451	1024	SEMIDO, TAD WORD	
989.	927	7452	3027	DCA CAD	/SAVE LOWER LIMIT IN CAD
990.	928	7453	5760	JMF I PREAD	/GET NEXT COMMAND
991.	929			/ Q HANDNANDLER	
992.	930			/PUNCH FIELD	
993.	931	7454	1031	QDO, TAD OCADF	
994.	932	7455	3030	DCA CADF	
995.	933	7456	7602	CLA HLT	/WAIT FOR USER TO TURN ON PUNCH
996.	934	7457	1030	TAD CADF	/GET FHELD
997.	935	7460	1357	TAD BP30C	/FLAG AS FIELD SPEC
998.	936	7461	4056	CALL	/PUNCH IT
999.	937	7462	6102	TYPE	
1000.	938	7463	5760	JMF I PREAD	/GET NEXT COMMAND
1001.	939			/ E, T HANDLERS	
1002.	940			/PUNCH CHECKSUM AND TRAILER	
1003.	941	7464	7602	EDO, CLA HLT	/WAIT FOR USER TO TURN ON PUNCH
1004.	942	7465	1046	TAD CHKSUM	/GET CHECKSUM
1005.	943	7466	7100	CLL	
1006.	944	7467	4056	CALL	/PUNCH IT
1007.	945	7470	7526	PUNCH	
1008.	946	7471	1353	TDO, TAD BM100	/INIT COUNT
1009.	947	7472	3020	ECA TEMP	
1010.	948	7473	1356	TDO1, TAD BP200	/PUNCH A 1 IN CHANNEL 8
1011.	949	7474	4056	CALL	
1012.	950	7475	6102	TYPE	
1013.	951	7476	2020	ISZ TEMP	/CCUNT OVER
1014.	952	7477	5273	JMP TDO1	/NO - REPEAT
1015.	953	7500	3046	DCA CHKSUM	/YES - REINITIALIZE CHECKSUM
1016.	954	7501	5760	JMF I PREAD	/GET NEXT COMMAND
1017.	955			/ P HANDLER	
1018.	956			/PUNCH OUT MEMORY IN EIN FORMAT	
1019.	957			/LOWER LIMIT IN CAD, UPPER LIMIT IN WORD	
1020.	958	7502	1031	PDO, TAD OCADF	
1021.	959	7503	3030	DCA CADF	
1022.	960	7504	7602	CLA HLT	/WAIT FOR USER TO TURN ON PUNCH
1023.	961	7505	1027	TAD CAD	/GET STARTING ADDRESS
1024.	962	7506	712C	STL	/PUNCH IT - FLAG AS ADDRESS
1025.	963	7507	4056	CALL	
1026.	964	7510	7526	PUNCH	
1027.	965	7511	4056	PDO1, CALL	/GET CONTENTS OF LOC
1028.	966	7512	6431	GETCAD	
1029.	967	7513	7100	CLL	/PUNCH IT

1030.	968	7514	4056	CALL	
1031.	969	7515	7526	PUNCH	
1032.	970	7516	1027	TAD CAD	/COMPARE CURRENT LOC
1033.	971	7517	7041	CIA	
1034.	972	7520	1024	TAD WORD	/AND HIGH LIMIT
1035.	973	7521	7650	SNA CLA	/EQUAL?
1036.	974	7522	5760	JMF I PBREAD	/YES - GET NEXT COMMAND
1037.	975	7523	2027	ISZ CAD	/NC - POINT A T AT NEXT LOCATION
1038.	976	7524	5311	JMF PDO1	/CONTINUE LOOP
1039.	977	7525	5303	JMP PDO+1	/GC AROUND TOP OF MEMORY -
1040.	978				/ PUNCH NEW NROGIN
1041.	979			/SUBROUTINE TO PUNCH A NUNEER	
1042.	980			/PUNCH NUMBER IN AC IN BIN FORMAT	
1043.	981			/SET CHANNEL 7 (TO FLAG AN ADDRESS) IF LINK IS SET	
1044.	982	7526	3020	PUNCH, DCA TEMP	/SAVE NUMBER
1045.	983	7527	1020	TAD TEMP	/GET HIGH HALF OF NUMBER
1046.	984	7530	7012	RTR	
1047.	985	7531	7012	RTR	
1048.	986	7532	7012	RTR	
1049.	987	7533	0355	AND BP177	/ISOLATE NUMBER AND LINK
1050.	988	7534	4056	CALL	/OUTPUT IT
1051.	989	7535	7543	PUNCH1	
1052.	990	7536	1020	TAD TEMP	/GET LOW HALF OF NUMBER
1053.	991	7537	0354	AND BP77	
1054.	992	7540	4056	CALL	/OUTPUT IT
1055.	993	7541	7543	PUNCH1	
1056.	994	7542	5461	RETURN	/RETURN
1057.	995			/SUBROUTINE TO PUNCH AC AND ACCUMULATE CHECKSUM	
1058.	996	7543	3021	PUNCH1, DCA TEMP2	/SAVD NUMBER
1059.	997	7544	1046	TAD CHKSUM	/ADD IT TO CHECKSUM
1060.	998	7545	1021	TAD TEMP2	
1061.	999	7546	3046	DCA CHKSUM	
1062.	1000	7547	1021	TAD TEMP2	/TYPE IT OUT
1063.	1001	7550	4056	CALL	
1064.	1002	7551	6102	TYPE	
1065.	1003	7552	5461	RETURN	/RETURN
1066.	1004			/ *** CONSTANTS	
1067.	1005	7553	7700	BM100, -100	
1068.	1006	7554	0077	BP77, 77	
1069.	1007	7555	0177	BP177, 177	
1070.	1008	7556	0200	BP200, 200	
1071.	1009	7557	0300	BP300, 300	
1072.	1010	7560	6200	PBREAD, READ	
1073.	1011			PAUSE	
1074.	1012		0020	LAST=20	
1075.	1013		0066	LT=66	
1076.	1014		0067	FIRST=67	
1077.	1015		0070	SEC=70	
1078.	1016		0071	THIRD=71	
1079.	1017		0072	DATA2=72	
1080.	1018		0073	PC2=73	
1081.	1019		0053	HOLD=53	
1082.	1020		0054	SAVFC=54	
1083.	1021				
1084.	1022		7576	*START+1576	
1085.	1023	7576	4056	CALL	
1086.	1024	7577	6400	CRFP	/ PUNCH OUT A CARRAGE RETURN
1087.	1025				/ AND LINEFEED AT THE START
1088.	1026				/ OF BIN
1089.	1027				
1090.	1028				

1091.	1029	7600		*START+1600		
1092.	1030					
1093.	1031	7600	7340	BIN,	CLA CLL CMA	/ SET LT TO 7777
1094.	1032	7601	3066		DCA LT	
1095.	1033	7602	3046		DCA CHKSUM	
1096.	1034	7603	3067		DCA FIRST	
1097.	1035	7604	3070		DCA SEC	
1098.	1036	7605	7340		CLA CLL CMA	
1099.	1037	7606	3054		DCA SAVPC	
1100.	1038	7607	1354		TAD K200	
1101.	1039	7610	3020		DCA LAST	
1102.	1040	7611	1357		TAD K102	
1103.	1041	7612	3067		DCA FIRST	
1104.	1042					
1105.	1043	7613	7340	BEGG,	CLA CLL CMA	
1106.	1044	7614	3072		DCA DATA2	
1107.	1045					
1108.	1046	7615	3073	BEG,	DCA PC2	
1109.	1047					
1110.	1048	7616	4056		CALL	
1111.	1049	7617	6463		LISN	
1112.	1050	7620	3053		DCA HOLD	
1113.	1051					
1114.	1052	7621	1053		TAD HOLD	
1115.	1053	7622	1356		TAD KRUB	
1116.	1054	7623	7700		SMA CLA	
1117.	1055	7624	5257		JMP RUM	
1118.	1056	7625	1053		TAD HOLD	
1119.	1057	7626	1352		TAD KCH8	
1120.	1058	7627	7650		SMA CLA	
1121.	1059	7630	5325		JMP LTC	
1122.	1060					
1123.	1061	7631	1053		TAD HOLD	
1124.	1062	7632	3020		DCA LAST	
1125.	1063					
1126.	1064	7633	1066		TAD LT	
1127.	1065	7634	7640		SZA CLA	
1128.	1066	7635	5215		JMP BEG	
1129.	1067					
1130.	1068	7636	1053		TAD HOLD	
1131.	1069	7637	1355		TAD KFD	
1132.	1070	7640	7700		SMA CLA	
1133.	1071	7641	5215		JMP BEG	
1134.	1072					
1135.	1073	7642	1053		TAD HOLD	
1136.	1074	7643	1046		TAD CHKSUM	
1137.	1075	7644	3046		DCA CHKSUM	
1138.	1076					
1139.	1077	7645	1067		TAD FIRST	
1140.	1078	7646	0353		AND KLONG	
1141.	1079	7647	7640	KLING,	SZA CLA	
1142.	1080	7650	5314		JMP PCL	
1143.	1081					
1144.	1082	7651	2073		ISZ PC2	
1145.	1083	7652	5272		JMP MORE	
1146.	1084					
1147.	1085	7653	1054	PCL2,	TAD SAVPC	
1148.	1086	7654	1070		TAD SEC	
1149.	1087	7655	3054		DCA SAVPC	
1150.	1088	7656	5306		JMP BACK	
1151.	1089					

1152.	1090	7657	4056	RUM,	CALL
1153.	1091	7660	6463		LISN
1154.	1092	7661	3053		DCA HOLD
1155.	1093	7662	1053		TAD HOLD
1156.	1094	7663	4056		CALL
1157.	1095	7664	6102		TYPE
1158.	1096	7665	1053		TAD HOLD
1159.	1097	7666	1356		TAD KRUB
1160.	1098	7667	7700		SMA CLA
1161.	1099	7670	5216		JMP BEG+1
1162.	1100	7671	5257		JMP RUM
1163.	1101				
1164.	1102	7672	2072	MORE,	ISZ DATA2
1165.	1103	7673	5277		JMP DL2
1166.	1104				
1167.	1105	7674	1053		TAD HOLD
1168.	1106	7675	3071		DCA THIRD
1169.	1107	7676	5215		JMP BEG
1170.	1108				
1171.	1109	7677	1067	DL2,	TAD FIRST
1172.	1110	7700	7002		BSW
1173.	1111	7701	1070		TAD SEC
1174.	1112	7702	3454		DCA I SAVPC
1175.	1113	7703	7000		NOP
1176.	1114	7704	2054		ISZ SAVPC
1177.	1115	7705	7300		CLA CLL
1178.	1116				
1179.	1117	7706	7300	BACK,	CLA CLL
1180.	1118	7707	1071		TAD THIRE
1181.	1119	7710	3067		DCA FIRST
1182.	1120	7711	1053		TAD HOLE
1183.	1121	7712	3070		DCA SEC
1184.	1122	7713	5213		JMP BEGG
1185.	1123				
1186.	1124	7714	1067	PCL,	TAD FIRST
1187.	1125	7715	0351		AND PMSK
1188.	1126	7716	7002		BSW
1189.	1127	7717	3054		DCA SAVEC
1190.	1128	7720	3067		DCA FIRST
1191.	1129	7721	1053		TAD HOLD
1192.	1130	7722	3071		DCA THIRD
1193.	1131	7723	7040		CMA
1194.	1132	7724	5215		JMP BEG
1195.	1133				
1196.	1134	7725	3066	ITC,	DCA LT
1197.	1135	7726	1020		TAD LAST
1198.	1136	7727	1352		TAD KCH8
1199.	1137	7730	7650		SNA CLA
1200.	1138	7731	5213		JMP BEGG
1201.	1139	7732	1067		TAD FIRST
1202.	1140	7733	0351		AND PMSK
1203.	1141	7734	7002		BSW
1204.	1142	7735	1070		TAD SEC
1205.	1143	7736	7041		CIA
1206.	1144	7737	1046		TAD CHKSUM
1207.	1145	7740	7041		CIA
1208.	1146	7741	3046		DCA CHKSUM
1209.	1147	7742	1067		TAD FIRST
1210.	1148	7743	0351		AND PMSK
1211.	1149	7744	1046		TAD CHKSUM
1212.	1150	7745	1070		TAD SEC

```

1213. 1151 7746 3047 DCA ACSAVE
1214. 1152 7747 5750 JMP I .+1
1215. 1153 7750 6727 ADO
1216. 1154
1217. 1155
1218. 1156 7751 0077 PMSK, 0077
1219. 1157 7752 7600 KCH8, 7600
1220. 1158 7753 0100 KLONG, 0100
1221. 1159 7754 0200 K200, 0200
1222. 1160 7755 7500 KFD, 7500
1223. 1161 7756 7401 KRUB, 7401
1224. 1162 7757 0102 K102, 0102
1225. 1163
1226. 1164
1227. 1165 7772 *START+1772
1228. 1166
1229. 1167 7772 7301 CLA CLL IAC / SET THE AC TO 0001
1230. 1168 7773 6400 6400 / CLEAR THE INTERCEPT JR.
1231. 1169 / DISPLAY
1232. 1170 7774 6402 6402 / TURN OF THE CP TIMER
1233. 1171 7775 5776 JMP I .+1 / GO TO THE START OF THE
1234. 1172 7776 6000 INIT / PROGRAM
1235. 1173
1236. 1174 7777 5372 JMP .-5 / ENTRY POINT FOR CP REQUEST
1237. 1175
1238. 1176
1239. 1177
1240. 1178
1241.
1242. END OF PASS 2
1243.
1244. 0 ERRORS DETECTED
1245. SYMBOL TABLE
1246.
1247. ACSAVE 0047 ACTEMP 0063 ADDR1 6737 ADDR2 674C ADDR4 6741 ADDR 0040 ADO 6727 AP250 7361
1248. AP256 7362 AP257 7363 AE260 7364 BACK 7706 BADD 6715 BDO1 7231 BDO 7215 BEGC 7613
1249. BEG 7615 BIN 7600 BKPT 7236 ELIST 6256 BM100 7553 BP177 7555 EP200 7556 BP300 7557
1250. BF77 7554 CADP 0030 CAD 0027 CALL1 0056 CALL 4056 CDF0 7141 CDO 7000 CFLAG1 6167
1251. CFLAG3 6177 CHKSUM 0046 CLOSE 6411 COMM 660C CON50 7007 CONT1 7033 CONT2A 7073 CONT2 7047
1252. CONT3A 7114 CONT3 7076 CONT4 7121 CONT5 740C CONT C035 CRDO 6616 CRLF 6400 DATA2 0072
1253. DFSAVE 0052 DFSET1 0066 DFSET 0073 EF1 0041 DL2 7877 DOTD0 6654 EDO 7464 ERROR 6337
1254. FIRS7 0067 FM10 7142 FP177 7143 FP2000 7146 FP2C0 7144 FP4000 7147 FP400 7145 FP7000 7150
1255. FP7400 7151 FP7770 7152 F1A 0000 F1 000C GDO 7200 GETCAE 6431 GETCD1 6437 GOADR 0043
1256. HOLD 0053 IADDR 7137 IFSAVE 0036 IFSET 0072 INIT1 6040 INIT 6000 INS 0037 JADR1 0025
1257. JADR 0042 KCH8 7752 KCRA 6056 KCPB 6057 KEEP 0045 KFD 7755 KLING 7647 KLONG* 7753
1258. KRUB 7756 KTY 6060 K0377 6336 K102 7757 K20C 7754 LAST 0020 LFD01 6645 LFO 6625
1259. LIMH1 0055 LIMLO 0054 LISN 6463 ESAVE 005C LTABL 6257 LTC 7725 LT 0066 MASK 0053
1260. MDO 6730 MORE 7672 MQSAVE 0051 M4 6134 NCONT 0044 NEWINS 0067 NTYPED 6423 OCADF 0031
1261. QDTLOC 7371 PBCDF 7446 PBCIF 7445 PBRAD 756C PCL2 7853 PCL 7714 PCNT5 7140 PCNT0 7366
1262. PCONT5 7367 PCREAD 6044 PC2 0073 PE8T1 7136 PDO1 7511 PDQ 7502 PFIELD 7347 PGTCD1 6461
1263. PIFSET 7135 PIFSTA 7365 PMSK 7751 PNUM2 6114 PNUM 6107 PNWINS 7134 POPJ 6072 PRAD 737C
1264. ESTCD1 6462 PUNCH1 7543 EUNCH 7526 PUSJ 6061 PZPAT 7450 P240 6137 P260 6140 P7 6135
1265. QDO 7454 RAM1 6051 RAM2 6052 RAM3 6053 READ1 6204 READ2 6235 READ3 6240 READ 620C
1266. RETRN1 0061 RETURN 5461 BUART 6160 RUM 7657 SAVPC C054 SCHAR 0025 SEC 0070 SEMIDO 7451
1267. SETCAD 6442 SETCD1 6451 SFLAG1 6166 SFLAG3 6176 SHUT 0032 SKIP3 6172 SKIP4 6173 SKPR 6162
1268. SKPTRB 6163 SLDO1 6610 SLDO 6601 SM270 6332 SM5 6333 SPNTEB 0026 SP10 6334 SP177 6742
1269. SP200 6743 SP215 6744 SP257 6745 SP260 6746 SP277 6335 SP400 6747 SP6201 6750 SP7600 6751
1270. STACK1 6050 STACK1 0074 STACK 0062 START 6000 SUB1 6045 SUB2 6046 SUB3 6047 TAELE1 6260
1271. TABLE2 6303 TDO1 7473 IDO 7471 TEMP2 0021 TEMP 0020 THRD 0071 TH5 6455 TOE 0023
1272. TP212 6456 TP215 6457 TP6201 6460 TRADF 0034 TRAD 0033 TRAP1 7443 TRAP2 7444 TRAP3 6054
1273. TRAP4 6055 TRAP 7447 TTYM 6477 TYE 6102 UADQ1 6712 UADO 6670 WCRA 6165 WCRB 6175
1274. WDO1 7314 WDO2 7337 WDO 7303 WORD 0024 WTTY 6171 WUART 6161 WVR 6174 ZPAT1 0064
1275. ZPAT 0005

```